applied sciences

MDPI

Article

A Novel Approach to SPAM Detection in Social Networks-Light-ANFIS: Integrating Gradient-Based One-Sided Sampling and Random Forest-Based Feature Clustering Techniques with Adaptive Neuro-Fuzzy Inference Systems

Oğuzhan Çıtlak [1,2], İsmail Atacak [2,3,*] and İbrahim Alper Doğru [2,3]

[1] Department of Computer Engineering, Graduate School of Natural and Applied Sciences, Gazi University, Ankara 06500, Turkey; oguzhan.citlak@gazi.edu.tr
[2] IoTLab, Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara 06560, Turkey; iadogru@gazi.edu.tr
[3] Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara 06560, Turkey
* Correspondence: iatacak@gazi.edu.tr

Abstract

With today's technological advancements and the widespread use of the Internet, social networking platforms that allow users to interact with each other are increasing rapidly. The popular social network X (formerly Twitter) has become a target for malicious actors, and spam is one of its biggest challenges. The filters employed by such platforms to protect users struggle to keep up with evolving spam techniques, the diverse behaviors of platform users, the dynamic tactics of spam accounts, and the need for updates in spam detection algorithms. The literature shows that many effective solutions rely on computationally expensive methods that are limited by dataset constraints. This study addresses the spam challenges of social networks by proposing a novel detection framework, Light-ANFIS, which combines ANFIS with gradient-based one-side sampling (GOSS) and random forest-based clustering (RFBFC) techniques. The proposed approach employs the RFBFC technique to achieve efficient feature reduction, yielding an ANFIS model with reduced input requirements. This optimized ANFIS structure enables a simpler system configuration by minimizing parameter usage. In this context, dimensionality reduction enables a faster ANFIS training. The GOSS technique further accelerates ANFIS training by reducing the sample size without sacrificing accuracy. The proposed Light-ANFIS architecture was evaluated using three datasets: two public benchmarks and one custom dataset. To demonstrate the impact of GOSS, its performance was benchmarked against that of RFBFC-ANFIS, which relies solely on RFBFC. Experiments comparing the training durations of the Light-ANFIS and RFBFC-ANFIS architectures revealed that the GOSS technique improved the training time efficiency by 38.77% (Dataset 1), 40.86% (Dataset 2), and 38.79% (Dataset 3). The Light-ANFIS architecture has also achieved successful results in terms of accuracy, precision, recall, F1-score, and AUC performance metrics. The proposed architecture has obtained scores of 0.98748, 0.98821, 0.99091, 0.98956, and 0.98664 in Dataset 1; 0.98225, 0.97412, 0.99043, 0.98221, and 0.98233 in Dataset 2; and 0.98552, 0.98915, 0.98720, 0.98818, and 0.98503 in Dataset 3 for these performance metrics, respectively. The Light-ANFIS architecture has been observed to demonstrate performance above existing methods when compared with methods in studies using similar datasets and methodologies based on the literature. Even in Dataset 1 and Dataset 3, it achieved a slightly better performance in terms of confusion matrix metrics compared to current deep learning (DL)-based hybrid and fusion methods, which are known as high-performance complex models in this field.

check for updates

Academic Editor: Rui Araújo

Received: 26 July 2025
Revised: 1 September 2025
Accepted: 12 September 2025
Published: 14 September 2025

Citation: Çıtlak, O.; Atacak, İ.; Doğru, İ.A. A Novel Approach to SPAM Detection in Social Networks-Light-ANFIS: Integrating Gradient-Based One-Sided Sampling and Random Forest-Based Feature Clustering Techniques with Adaptive Neuro-Fuzzy Inference Systems. *Appl. Sci.* **2025**, *15*, 10049. https://doi.org/10.3390/app151810049

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Additionally, the proposed model not only exhibits high performance but also features a simpler configuration than structurally equivalent models, providing it with a competitive edge. This makes it a valuable for safeguarding social media users from harmful content.

Keywords: social networks; spam detection; adaptive neuro-fuzzy inference system; gradient-based one-side sampling; RF-based clustering

1. Introduction

Modern technological advancements have led to significant developments in internet usage. The increasing prevalence of social media platforms, which enable people to communicate regardless of time or location, is one notable outcome of this progress. This proliferation significantly affects communication and information exchange. Social networking platforms, including X (Twitter), Facebook, LinkedIn, Instagram, TikTok, and WhatsApp, are extensively used by millions of individuals [1]. The dissemination of information on these widely employed social platforms, along with their accessibility to various content, renders them susceptible to exploitation by nefarious individuals. The absence of security threats in these networks, the maintenance of their quality, and users' demand for safety underline the need to detect spam on social networks to combat malevolent individuals [2]. In social networks, "spam" is usually characterized as undesirable conduct that irritates users and diminishes the platform's overall quality [3]. The deterioration of user trust, the decrease in the platform's reputation, and the emergence of security threats deter users and prompt their departure from these platforms. Moreover, spam identification on social networks constitutes a technical challenge and a crucial element for the platforms' long-term success. These platforms must guarantee that their users are engaging in a secure environment. Consequently, it is imperative for social networking platforms to develop and implement their own efficient spam detection systems for their continued viability. Also, the ongoing advancement and enhancement of social network spam detection systems must persist [3,4].

On X, a leading social networking platform, new developments in spam detection point out the need for new strategies to combat spam. In particular, integrating techniques such as Artificial Intelligence (AI) and Machine Learning (ML) into new approaches significantly improves spam identification [5–8]. However, the literature shows that although many studies have been performed on spam detection on X, these studies are insufficient. The prevalent spam detection methodologies, listed alphabetically, include Account Creation Time-Based Method (ACTM) [9], Anomaly Detection Method (ADM) [10,11], Automated Systems (AS) [12,13], Behavioral Analysis Approaches (BAA) [14,15], Comparison and Contrastive Approaches (CCA) [16,17], Content-Based Filtering (CF) [18], Deceptive Information Detection Method (DIDM) [19,20], Deep Learning Methods (DLM) [21–23], Ensemble Learning Method (ELM) [12,24], Following and Follower Comparison Method (FFCM) [25], Geolocation Analysis Technique (GAT) [26], Honeypot-Based Spam Detection (HSD) [27], Link Analysis Approach (LAA) [23,28,29], Machine Learning Methods (MLM) [29–31], Natural Language Processing Methods (NLPM) [32,33], Spammer Detection Tools (SDT) [34,35], Trend-Topics Analysis Method (T-TAM) [7,36,37], User Reports Methods (URM) [3,38], and Using Blacklist (UB) [39]. These methods are employed for various purposes, including advertising, phishing, malware dissemination, bot and fake account creation, and fraud [40–42]. Table 1 illustrates the frequently employed spam detection methodologies on X.

Table 1. The advantages and disadvantages of common spam detection methodologies on X.

No	Common Spam Detection Methods	Advantages							Disadvantages						
		Quick Spam Detection	No Need Complex Algorithms	Can Work Dynamically	Can Include Many Methods	Flexible and Adaptive Models	Fast and Effective Blocking	Real-Time Detection	High Rate of False Positives	High Processing Power Required	Spammers Can Easily Bypass This Method	Need to Learn	Up-to-Date and Maintenance Cost Highly	Complex	The Overfitting Issue
1	Account Creation Time-based Method (ACTM)	✓	✓						✓		✓				
2	Anomaly Detection Method (ADM)	✓		✓					✓	✓		✓			
3	Automated Systems (AS)	✓				✓	✓	✓					✓	✓	
4	Behavioral Analysis Approaches (BAA)										✓				
5	Comparison and Contrastive Approaches (CCA)				✓				✓					✓	
6	Content-based Filtering (CF)		✓						✓						
7	Deceptive Information Detection Method (DIDM)								✓		✓	✓			
8	Deep Learning Methods (DLM)				✓	✓	✓	✓		✓				✓	✓
9	Ensemble Learning Method (ELM)				✓					✓		✓		✓	✓
10	Following and Follower Comparison Method (FFCM)	✓	✓								✓				
11	Geolocation Analysis Technique (GAT)			✓					✓		✓				
12	Honeypot-based Spam Detection (HSD)				✓			✓				✓		✓	
13	Link Analysis Approach (LAA)	✓	✓								✓				
14	Machine Learning Methods (MLM)	✓		✓	✓	✓	✓	✓				✓			✓
15	Natural Language Processing Methods (NLPM)				✓	✓				✓	✓				
16	Spammer Detection Tools (SDT)			✓				✓	✓						
17	Trend-Topics Analysis Method (T-TAM)	✓								✓					
18	User Reports Methods (URM)	✓	✓	✓											
19	Using Blacklist (UB)						✓	✓			✓		✓		

Since spamming activities frequently use newly created accounts, the ACTM spam detection approach enables rapid identification of spam. This method does not require sophisticated calculations since it relies solely on the account creation timestamp. This approach may incorrectly classify innocent people who establish new accounts as spam due to its temporal nature. Malicious individuals can sustain the activity of spam accounts they establish for an extended duration to evade this strategy. In addition, regular users who have previously established an account may be disregarded by this method when they exhibit spamming behavior. ADM can quickly identify user accounts that differ from normal activity and classify them as spam. It is a dynamic method that requires learning

to improve accuracy in decisions. Nonetheless, it is prevalent when typical conduct is wrong-identified as suspicious activity and classified as spam. High processing power is needed to accomplish the demand for learning. The necessity for precise and high-quality data may result in high false-positive rates for this strategy. ML and AI algorithms in AS can rapidly examine large amounts of accurate and high-quality data autonomously, enabling spam detection. Nonetheless, the installation, maintenance, and updates of such systems can be expensive. Data security breach laws in countries can also impact the continuous monitoring of user data. Zhang et al. [43] obtained a high value compared to other Naive Bayes, SVM, BiLSTM, BERT-base, and BERT + CNN models in a proposed study on AI-based spam detection and classification. Using a multilingual structure in the dataset used in this study does not necessarily mean that high success will be achieved. However, it can be optimized for real-time applications because of its minimal latency. The CCA approach is a little complex and encompasses numerous techniques. One of the techniques involves analyzing spam and non-spam users through a categorization technique employed in ML. The analysis of the relationships inside the messages sent by users is particularly notable. Examining several users within a methodology can introduce complications. Such issues may potentially result in math errors and a loss of integrity. The CF technique attempts to identify spam by directly examining tweet content. However, the erroneous classification of legitimate content as spam represents a significant challenge frequently encountered by this technique. The DIDM approach closely resembles the ADM because both require a learning process. It can achieve high accuracy in identifying fraudulent information, but legitimate content may also be incorrectly classified as deceptive. The DLM approach can yield good outcomes when trained on extensive datasets. This approach can autonomously extract features from the data, diminishing the necessity for human feature extraction. However, processing massive datasets necessitates substantial computational capability and time. Overfitting issues and training data are among the most prevalent concerns. In addition to this model, the DLM approach integrates numerous models. A comprehensive literature study was carried out, examining over thirty artificial intelligence techniques in addition to this model. In general, hybrid models using ANN + ANFIS + Genetic Algorithms together have been claimed to be successful in high accuracy predictions and error detection in another study by Soto-Diaz et al. [44]. The ELM approach can attain elevated accuracy rates by integrating the outcomes of multiple methods. Due to its inclusion of several models, its generalization capability is quite effective. Nonetheless, training and processing several models necessitates substantial computer capacity. This strategy requires a substantial quantity of labeled data to be effective. The combination of many methodologies may render model management challenging and complex. The FFCM approach is a spam detection model that performs simple calculations by comparing the ratios between the followers and following accounts of X users. The differences between follower and following accounts are quickly identified. However, this method, which relies solely on follower and following counts, is very limited in its application. However, malicious individuals can easily bypass this method. In a study by Shifath et al. [45], techniques such as RoBERT and ALBERT are regarded as the most probable classifications among modern techniques employed for spam detection. With these techniques, models like ANFIS may improve the accuracy of essential predictions. The GAT approach uses tweet location data to determine whether spam accounts are in specific regions; however, legitimate users may have their tweets positioned in certain geographic areas. This constitutes a significant issue encountered by the GAT approach. Furthermore, most of the tweets lack locational data. In the HSD approach, malicious users disseminating spam waste their time and resources while attempting to divert attention from their actual goals. Nonetheless, the capacity to identify certain categories of spam, learning impairments, and

management challenges constitute some of the most important disadvantages of the HSD technique. The LAA technique considers linkages within tweets. Tweet links that direct to harmful websites and include sensitive content are classified as spam. This method is quite constrained, like the FFCM, as it solely examines links. The MLM method is one of the most frequently used for spam detection on many social platforms. High accuracy, learning, and adaptability with other spam detection models are the most important advantages it has, but the need for a lot of labeled data (like the ELM) and overfitting issues (like the DLM method) are major problems. The NLPM approach tries to identify spam by examining the linguistic structure and content of tweets. Additionally, it can adjust to novel and evolving spam detection techniques, such as the MLM method. In the area of NLP, word-based embedding techniques may not be preferred. The impact of techniques like BERT and ELMo (Embeddings from Language Models) on spam identification was investigated in this study by Alshattnawi et al. [46], which also evaluated the use of contextualized models within the framework of word-based embedding methods for spam detection in social networks. This study offers valuable information to promote the application of contextual embedding. Still, Twitter contains tweets composed in several languages, necessitating multilingual models. The need for substantial computing resources and annotated data presents a significant challenge. A major issue with the SDT approach is that actual users falsely classify it as spam. The T-TAM approach evaluates behaviors in trending subjects similarly to the ADM; however, data quality and processing capacity are critical requirements. The URM approach is comparatively economical and depends only on user reporting. Users usually avoid spam due to their suspicion regarding its existence. Although in the UB model, these lists are applicable and user-friendly, the failure to monitor upgrades and emerging spam tactics constitutes one of the most significant challenges. Although many methods have been proposed in the literature for spam detection, existing approaches generally have difficulty coping with high-dimensional data, exhibit poor performance in the face of unbalanced data distribution, and may be insufficient in terms of explanation. Furthermore, training models such as DLM, CCA, and NLPM involve significant computational charges and raise issues with efficiency in real-time applications. Therefore, there is a growing demand for lightweight and easily comprehensible spam detection systems.

Some of the reviewed advantages and limitations in studies cited as literature on X involve fast and structurally straightforward methodologies, although they have performance shortcomings that require refinement. Others excel in spam detection across specific datasets but introduce critical challenges, such as structural complexity, computational overhead, prolonged training duration, dependence on various large-scale datasets for reliable training, and susceptibility to overfitting. In this context, ANFIS can effectively solve some problems arising in this area because of its low computational cost, speed, interpretability advantages, and ability to work with small data groups. However, as the number of input variables increases in ANFIS architectures, important issues such as rule explosion, overfitting, and loss of interpretability arise, limiting their usability as spam detection models. This study proposes a novel spam detection model called Light-ANFIS, which combines the GOSS technique with the ANFIS architecture for a fast and effective training process by implementing effective feature reduction that minimizes the number of inputs using our developed RFBFC method. Its novelty lies in two key aspects: (1) RFBFC, our newly introduced method for efficient feature reduction, and (2) the synergistic use of GOSS to enable substantial data reduction without performance degradation. Below are the contributions of Light-ANFIS to academic literature.

1- The number of features effectively reduced by the proposed RFBFC technique also minimized the number of inputs used in ANFIS. This situation significantly reduced

the number of parameters used in the ANFIS based on fuzzy sets and rules, enabling a simpler ANFIS configuration.

2- Combining ANFIS with the RFBFC and GOSS methods enables a substantial reduction in feature columns and data samples within the training dataset while maintaining performance. The structural streamlining outlined in the first bullet point and the accompanying dimensionality reduction led to faster training duration in the proposed Light-ANFIS framework.

3- The study demonstrates that feature reduction using RFBFC and data sample reduction via GOSS-based selective data sampling not only maintains performance but also improves it in some datasets. Therefore, integrating these techniques with ANFIS could enhance its overall performance.

4- By integrating RFBFC and GOSS techniques into all fuzzy inference-based systems, significant contributions can be made in terms of structural simplicity, performance, and speed.

The remainder of this study is as follows: Section 2 presents studies based on the literature related to the topic. Section 3 details the datasets employed, introduces the Light-ANFIS architecture (including its components) and defines evaluation metrics. Section 4 presents ablation experiments, empirical analyses, and comparative benchmarks against existing studies and discusses performance outcomes. Section 5 synthesizes these findings and offers insights into future directions.

2. Related Studies in the Literature

Numerous studies exist regarding spam analysis and detection in social networks. This section reviews studies relevant to the proposed study, covering Machine Learning (ML), Fuzzy Inference Systems (FIS), and Deep Learning (DL)-based methodologies for spam detection. These studies are critical as they highlight how the proposed model addresses social network spam, a severe issue undermining user security and degrading platform experience. Some of the studies are presented below:

Alom et al. [47] developed a new unified model based on the CNN algorithm, using tweet text and user metadata to detect spam in social networks. They compared the performance of this method with methods including ML and DL techniques through experimental studies on accuracy, recall, precision, and F1-score performance metrics. They validated the proposed model using these techniques on the Twitter Social Honeypot and Twitter 1KS-10KN datasets. Using the proposed unified classifier, they achieved the highest accuracy of 99.68% on the Twitter Social Honeypot dataset and 93.12% on the Twitter 1KS-10KN dataset.

Agarwal et al. [48] proposed a spam detection model that analyzes spam messages and emails to address data shortcomings and limited data problems, targeting high accuracy using NLP and AMALS (Approximations with Modifying and Alternating Least Squares) models. The model implemented the natural language processing procedures of Bag-of-Words and Count Vectorizer, together with the sparse data processing methods, including feature extraction. Missing data was estimated using the AMALS model, whereas spam and non-spam classifications were performed by machine learning-based methods. The quality of NLP-based spam filtering outperforms that of traditional TF-IDF algorithms in the literature, achieving 98% accuracy. The study exclusively assessed classical machine learning algorithms, without any comparison to deep learning or transformer-based methods such as BERT and GPT.

Pal et al. [8] present a Transformer-based methodology employing NLP techniques for the detection of fake news. This study showed the better accuracy of transformer-based models, despite the common usage of RNN, CNN, and classical approaches in the literature.

The performance comparison of the proposed model reveals the following metrics: For RoBERTa, accuracy is 97.853%, precision is 98.104%, recall is 97.502%, and the F1 score is 97.802%; for SVM, accuracy is 95.452%, precision is 95.803%, recall is 95.102%, and the F1 score is 95.451%; for logistic regression, accuracy is 93.203%, precision is 93.453%, recall is 92.953%, and the F1 score is 93.202%. The Light-ANFIS-based model offered in our study provides better metrics compared to the proposed transformer-based model.

Dhar and Bose [21] proposed an ensemble DL model to classify Twitter spam efficiently. Their research addressed three core challenges: pinpointing critical features, assessing the practicality of various model architectures, and enhancing computational speed. They implemented feature engineering strategies, including the Rich Feature Set (RFS), Naive Feature Set (NFS), and Deep Feature Set (DFS), to structure their approach. The experiments demonstrated that DFS and RFS surpassed NFS in terms of classification accuracy (measured using confusion matrices), whereas NFS excelled in processing speed. Through empirical comparisons of ML and DL models, they designed a robust ensemble DL system that maintained high accuracy and outperformed alternatives in terms of speed and predictive performance.

Shifath et al. [45] introduced a novel DL ensemble model for Covid19 fake news detection in English at Constraint@AAAI2021. Their model integrates predictions from eight state-of-the-art language models, such as GPT-2, XLNet, BERT, RoBERTa, Distil-RoBERTa, ALBERT, DeBERTa, and BART, via a meta-classifier architecture comprising fully connected layers, linear layers, and a softmax classification output layer. Evaluated on Covid19 datasets, this approach demonstrated exceptional performance metrics: accuracy (97.9906542%), precision (97.9913119%), recall (97.9906542%), and F1-score (97.9907901%).

Altwaijry et al. [49] performed a study to determine the performance of several deep learning models to recognize phishing emails, employing models such as RNN, LSTM, GRU, and CNN with NLP techniques to classify phishing, legitimate, and spam emails over different datasets. The proposed 1D-CNNPD deep learning method obtains an accuracy of 98.87% and an F1 score of 93.38%. The proposed method has considerable computational costs and requires substantial GPU/TPU resources for practical ap-plication.

Wang et al. [50] proposed a Deep Contrastive Graph Clustering (BotDCGC) framework, which is an unsupervised social bot detection, to tackle Twitter bot-related challenges. Their approach integrates a graph attention encoder with an inner product decoder, which was validated experimentally on the Twibot-20 and Cresci-2015 datasets. The results showed superior performance—80.95%, 87.31%, 78.29%, 78.88% accuracy, precision, recall, F1 (Twibot-20) and 93.34%, 93.35%, 92.26%, 92.75% (Cresci-2015), respectively—proving that their model outperforms.

Differing from existing methods, Li et al. [51] introduced a novel fuzzy-logic-driven web object categorization system to detect fake news. By utilizing DL algorithms for feature extraction, they employed high-dimensional indexing to rank features using relevance metrics. Subsequent application of Deep Fuzzy Learning enabled the adaptive assessment of web content integrity. Tests on a Python 3.8/Windows 10 setup (4 GB RAM, Intel I3) yielded robust results—95.46% accuracy, 95.45% F1-score, 96.73% precision, and 94.87% recall—demonstrating the model's efficacy in identifying fake news.

A study by Wani et al. [52] describes a framework combining NLP and deep learn-ing algorithms to identify artificial intelligence-generated spam, where the Word2Vec-based BiLSTM achieved an accuracy of 98.46%, a precision of 0.98, a recall of 0.97, and an F1 score of 0.98. The efficacy in identifying AI-generated spam was shown by an F2 score of 0.9810 and an F0.5 score of 0.9857. The Word2Vec-based LSTM model achieves 97.58% accuracy, 0.97 precision, 0.96 recall, and a 0.97 F1 score, while the Word2Vec-based CNN model obtains 97.61% accuracy, 0.97 precision, 0.96 recall, and a 0.97 F1 score. This study is

one of the initial comprehensive deep learning frameworks for artificial intelligence-based spam detection. The selection of text-only functionalities restricts this effort.

In a study on fake news identification, Nair et al. [53] developed a knowledge-based method. Their framework employs NLP, graph theory principles, and information retrieval techniques for automated fact-checking. The detection system incorporates four core attributes from knowledge bases: (1) subject-predicate-object (SPO) triples, (2) SPO sentiment polarity, (3) SPO occurrence, and (4) topic modeling. Performance evaluation of multiple DL architectures indicated that GRU networks achieved 75% accuracy, while LSTM models reached 79%. Interestingly, GPT-3 outperformed the others at 81%, whereas BERT showed relatively lower accuracy (61%) across benchmark datasets, including FakeNewsNet, BuzzFeedNews, LIAR, and user-generated opinions.

Jain et al. [54] proposed an ANFIS-based tweet recommendation system on Twitter to mitigate user-specific ambiguities and gaps by tailoring suggestions that are aligned with their interests. Their methodology integrates hybrid content analysis and collaborative filtering, identifying source/target user profiles and mapping interest correlations before constructing the ANFIS framework. The efficacy of the model was validated via RMSE (scoring an impressive 0.93), with an average test error of 0.027.

Suganthi and Prabha [55] employed Fuzzy Similarity-Based Hierarchical Clustering (FS-HC) method to detect communities on Twitter. The dataset, gathered via the Twitter API, underwent preprocessing using unigrams, bigrams, and 1–3 g. Features were extracted via Word Embeddings and Term Frequency-Inverse Document Frequency (TF-IDF), followed by standard preprocessing (stemming, stop-word removal, and tokenization). Their model used Fuzzy Similarity Matrices, Dendrograms, and Consensus Matrices, with performance benchmarked against prior study using conventional ML evaluation metrics. The proposed model demonstrated reliable performance, achieving 92% accuracy.

In another study by Rajesh et al. [56], the detection of spam comments employed ANFIS and NLP algorithms combined to identify spam comments. The study utilized several available datasets featuring labeled spam and non-spam comments. The methods for data preparation employing NLP, feature processing with ANFIS, and the design of fuzzy rules and membership functions were completed with ML classifiers. The ANFIS + SVM model proposed in the study offers better results relative to alternative models, achieving an accuracy of 92.5%, precision of 91.8%, recall of 93.2%, an F1 score of 92.5%, and a ROC-AUC of 0.95, in comparison to ANFIS + Logistic Regression, ANFIS + Random Forest, SVM (baseline), LR (baseline), and RF (baseline). This study's benefit comes from its capacity to effectively manage language confusion using NLP and ANFIS.

Gracia Bettyet al. [57] conducted a series of analyses using 20 different ML algorithms—such as Random Forest, Light Gradient Boosting Machine (LightGBM), and Logistic Regression—on a publicly available dataset to detect spam comments from YouTube videos. The dataset obtained from Kaggle contains 11,142 comments, with 5569 labeled as spam and 5573 as non-spam. Their two-phase study began with an in-depth evaluation that narrowed down to nine top-performing ML models. These nine algorithms were rigorously assessed in phase two based on accuracy metrics. LightGBM outperformed all others with a remarkable accuracy rate of 94%.

Gong and Liu [58] introduced a LightGBM-based model to detect botnets, which are persistent threats to network traffic, and benchmarked it against established ML models such as Random forest (RF), Bagging Classifier, XGBoost, Decision Forest, and K-nearest neighbor (KNN). Their evaluation employed the CTU-13 dataset. The performance was evaluated by confusion matrix metrics (recall, precision, accuracy, and F1-score) and training durations. The LightGBM model outperformed the others, with a 99.14% recall score and an 18 s training time. It also rivaled KNN in terms of accuracy at 96.62%.

During the general evaluation of these studies, the advantages revealed included the capability to combat the overfitting issue and difficulties that spammers face in overcoming the suggested methods due to their existing structure, as well as the minimal processing power required. Nonetheless, the necessity of understanding the proposed models for accurate classifications and generalizations, the resources invested in this process, and the complex arrangements of the models represent disadvantages. In addition to the two recommended studies [50,51], the capability of alternative models to detect spam faster and function dynamically via the integration of several models provides a considerable advantage in spam detection. Two studies with an elevated risk of false positives [54,58] offer a comparative advantage over others due to their fewer expenses for upgrades and maintenance.

When examining spam detection methods developed using DL techniques on social networks, it is generally observed that these models are structurally complex and computationally expensive. In addition, large-scale datasets with various data are required to achieve high performance with these models. Therefore, some studies have shown that specific DL models yield low performances. Spam detection analyses from the perspective of ML-based models have shown that effective traditional ML methods, such as KNN and ensemble-based ML methods, such as RF, XGBoost, and LightGBM, have been successfully employed. However, similar to DL methods, the performance of these approaches also declines in problems involving sparse and low-sample data. Recently, the ANFIS architecture, which effectively combines human learning capability with inference philosophy, has been successfully used in this field, albeit in a limited number of applications. The number of attributes applied to this system is the primary factor restricting the applicability of the ANFIS architecture in such applications. Expanding the inputs in ANFIS exponentially increases the parameter requirements, resulting in sluggish training speeds, which can sometimes render the system unusable. Large-scale training datasets face comparable challenges. This study introduces an innovative solution within the ANFIS framework by incorporating the RFBFC and GOSS techniques into the conventional architecture for spam detection. While the RFBFC technique effectively reduces data dimensionality using feature engineering, the GOSS technique achieves efficient training sample reduction on data basis.

3. Materials and Methods

This section details our custom dataset derived from Twitter user profiles, alongside two benchmark datasets used to validate the proposed method. The Light-ANFIS approach, which combines the GOSS and RFBFC methodologies, is comprehensively described, including all composite elements. The performance metrics used to evaluate the proposed method are presented at the end of this section.

3.1. About Datasets and Data Collection

The present study employs three different Twitter datasets—Dataset 1, Dataset 2, and Dataset 3—to evaluate the performance of the LightANFIS approach in detecting spam on social networks. The details of these datasets are explained in the subheadings provided below.

3.1.1. Dataset 1

The first dataset includes the data used for spam detection in a prior study, which we gathered for that research [40]. The researchers generated the labeled version of the dataset using the crowdsourcing approach on 1225 Twitter accounts retrieved via the API. This dataset consists of 11 features: User_Statuses_Count (USC), Sensi-

tive_Content_Alert (SCA), User_Favourites_Count (UFC), User_Listed_Count (ULC), Source_in_Twitter (SITW), User_Friends_Count (UFRC), User_Followers_Count (UFLC), User_Location (UL), User_Geo_Enabled (UGE), User_Default_Profile_Image (UDPI), and ReTweet (RTWT), along with one class tag. Feature values are non-numeric data categorized by a defined criteria confidence interval, such as "yes/no", "true/false", and "900–999". They generated a new dataset of 6125 instances with six continuous properties by implementing these values in the Fuzzy Inference System (FIS) models used in their research, utilizing several preprocessing techniques such as data parsing, multiplexing, transformation, and merging. This dataset has five features—USC, UFC, ULC, UFRC, and UFLC—that identify the forms of the labeled dataset transformed into real values, whereas one feature signifies the form of the labeled dataset that has been combined and converted to the actual value. This research used the preprocessed version of Dataset 1.

3.1.2. Dataset 2

The second dataset combines subsets from two studies: one focused on Twitter spam accounts [21], and the other examined fake profile detection [59,60] in social networks. It comprises 2818 user accounts—1480 legitimate users and 1338 spammers. During preprocessing, only 18 of the 35 features flagged with '✓' in Table 2 were retained; the remaining 17 were excluded because of their minimal impact on spam identification.

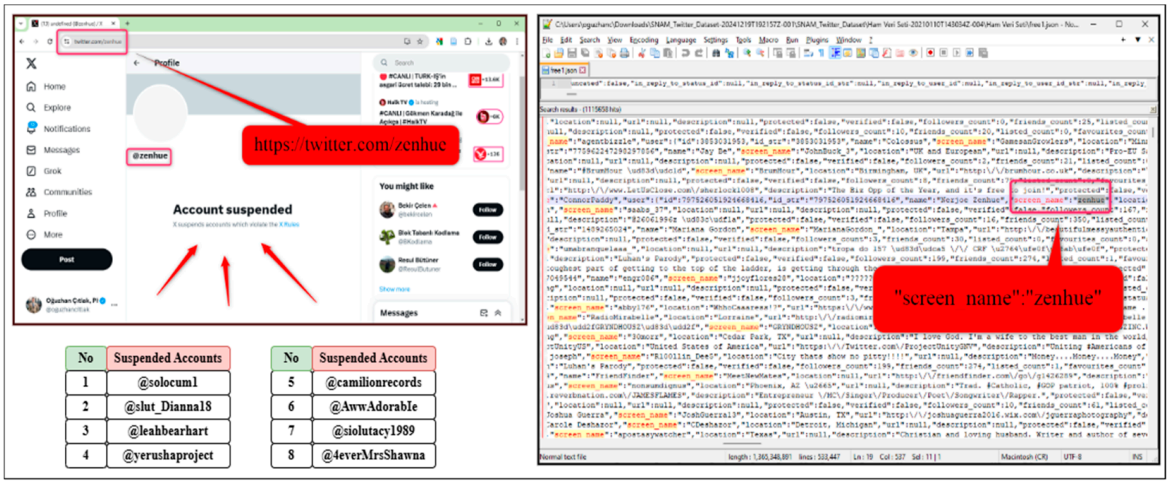
Table 2. Features included or omitted from the relevant datasets in Dataset 2.

No	Feature	Used	No	Feature	Used
1	id		19	profile_banner_url (PBU)	✓
2	name		20	profile_use_background_image (PUBG)	✓
3	screen_name (SRN)		21	profile_background_image_url_https (PBIA)	
4	fav_number (FVN)	✓	22	profile_text_color (PTC)	✓
5	statuses_count (STC)	✓	23	profile_image_url_https (PIH)	
6	followers_count (FOC)	✓	24	profile_sidebar_border_color (PSBC)	✓
7	friends_count (FRC)	✓	25	profile_background_tile (PBT)	✓
8	favourites_count (FAC)	✓	26	profile_sidebar_fill_color (PSFC)	
9	listed_count (LSC)	✓	27	profile_background_image_url (PBIU)	
10	created_at (CRT)		28	profile_background_color (PBGC)	
11	url	✓	29	profile_link_color (PRLC)	
12	lang		30	utc_offset (UOF)	
13	time_zone (TMZ)		31	Protected (PRTC)	
14	Location (LOC)	✓	32	Verified (VRF)	
15	default_profile (DFP)	✓	33	Description (DSC)	✓
16	default_profile_image (DPI)	✓	34	Updated (UPD)	
17	geo_enabled (GOE)	✓	35	dataset	✓
18	profile_image_url (PRIU)				

In Table 2, features 4, 5, 6, 7, 8, 9, and 20 contain numerical values, making them suitable for our proposed model; therefore, their existing values are preserved. For features 14, 15, 16, and 17, empty rows are marked as "0" and filled rows as "1." In feature 19, rows with "NULL" are marked as "1," while others are marked as "0." For feature 22, accounts with the default code value of "333333" are marked as "1" and the others as "0." In feature 33, empty rows are marked as "1," while others are marked as "0." For feature 25, the default values are marked as "0" and others as "1". In feature 35, "INT" values indicated for spam accounts are marked as "1," while "E13" values used for non-spam accounts are marked as "0." The normalization method is used for these operations, and the data are formatted in this step. Normalization includes converting text data to lower/upper case and standardizing date formats.

3.1.3. Dataset 3

Third, our custom dataset, comprising details of 22,096 user accounts, was gathered from Twitter's social platform using Python 3.8 and the Tweepy 4.8.0 library via its API. Data extraction via this method necessitates authentication through account-specific credentials: Consumer Key, Access Token, Consumer Secret, and Access Token Secret. Python was selected for scripting automated data collection because it eliminates the need for a standalone compiler, accelerates development time, and natively supports libraries that are important for Twitter developer workflows. The classes required for streaming and authentication in Tweepy are "Stream," "OAuthHandler," and "StreamListener." Parameters were obtained from the command line using Argparse, whereas String was used for character operations. The function between lines 1–3 in the pseudocode table of Algorithm 1 below acquires the -q (search query) and -d (data folder) parameters from the command line. If no query was provided, "-" was used as the default. In line 4, a customized class listens to live data from Twitter and writes it to the JSON file shown in Figure 1 in the specified format, ensuring that each new data entry is saved. As illustrated in the pseudocode table, any characters not considered valid in the data were replaced with "_". After line 12, the arguments are obtained from the command line. Live data streaming operations were executed using Twitter API keys (Consumer Key, Access Token, Consumer Secret, and Access Token Secret).



No	Suspended Accounts
1	@solocum1
2	@slut_Dianna18
3	@leahbearhart
4	@yerushaproject

No	Suspended Accounts
5	@camilionrecords
6	@AwwAdorable
7	@siolutacy1989
8	@4everMrsShawna

Figure 1. An example of an account suspended by Twitter, other accounts, and the shape of the dataset interface.

The raw dataset was acquired from the @oguzhancitlak account. According to its spam policy, Twitter suspends accounts depending on its ability to identify disruptive users, as seen in Figure 1.

Twitter later suspended the spam accounts included in Dataset 3. Searching the "screen_name" of these accounts on Twitter shows that the account has been suspended. Figure 1 illustrates the interface of the suspended "zenhue" account, other familiar suspended account names, and a snapshot of the initial raw dataset presented in the JSON format. Dataset 3 contains 22,096 samples associated with Twitter user accounts. Among them, 13,565 were classified as spam, while the remaining 8531 were related to innocent user data. Table 3 presents the features, classifications, scale ranges, and descriptions of the dataset [40,61].

Algorithm 1 Real-Time Twitter Data Collection via API

```
Input    query          // Keyword to track
         data_dir       // Directory to store collected data
         API_keys       // Twitter API credentials
Output   stream_<query>.json   // JSON file containing collected tweets
Begin
1.  def get_parser():
        // Parse command-line arguments to get query and data_dir
    return parser
2.  Authenticate with Twitter using API_keys
3.  Format query to create a valid filename
        // For each character in query:
    If the character is valid (letter, digit, -, _, .), keep it
    Otherwise, replace it with '_'
4.  class MyListener(StreamListener):
            // Define MyListener class to handle streaming:
5.          on_data(data):
6.          Append incoming tweet data to stream_<query>.json
            // The folder where the data will be saved was created in json format.
7.          Print data to console
8.          If error occurs, wait 5 seconds and continue
9.          on_error(status):
10.         Print error status to console
11. Initialize Twitter stream with MyListener(query, data_dir)
    Define on_error(status):
            // Print error status to console
            // Start streaming tweets filtered by query
12. Start Twitter stream:
    Filter tweets by query
    Use MyListener to handle incoming data
13. end
```

Table 3. Taxonomy and criteria ranges of Dataset 3.

No	Type in Model	Type	Evaluation Range	Explanation
1	USTC:User_Statuses_Count	Tweets	'20–99', '100–199', . . ., '1,000,000–1,999,999'	The tweet count posted by the user (including retweets).
2	USCA:Sensitive_Content _Alert	Array of Object	TRUE/FALSE	USCA refers to sensitive objects within a tweet's text or a user object's text fields.
3	UFVC:User_Favourites _Count	Boolean	'0–9', '10–19', '20–29', . . ., '100,000–1,999,999'	UFC is the number of Tweets a user account has liked over its lifetime.
4	ULSC: User_Listed_Count	Int	'0–9', '10–19', '20–29', . . ., '900–999'	ULSC indicates the count of public lists a user belongs to.
5	SITW: Source_in_Twitter	String	YES/NO	SITW reveals the utility of sending Tweets as an HTML-formatted string. Tweets originating from the Twitter website carry a web source value.
6	UFRC:User_Friends_Counts	Int	'0–9', '10–19', '20–29', . . ., '1000–99,999'	The user count followed by an account. Under specific circumstances, this field may temporarily appear as zero.

Table 3. *Cont.*

No	Type in Model	Type	Evaluation Range	Explanation
7	UFLC:User_Followers_Count	Int	'0–9', '10–19', '20–29', . . ., '100,000–1,999,999'	The tweet counts a user has liked over the account's lifetime.
8	ULOC:User_Location	String	YES/NO	ULOC displays a user-defined location for an account profile. The search service may sometimes interpret these fields ambiguously.
9	UGEO:User_Geo_Enabled	Boolean	TRUE/FALSE	When set to True, UGEO means that the user has allowed location tagging for tweets.
10	UDPI:User_Default_Profile _Image	Boolean	TRUE/FALSE	When UDPI is True, it implies that the user has not uploaded a custom profile picture, and the system is using a default image instead.
11	RTWT:ReTweet	Boolean	TRUE/FALSE	RTWT shows whether the authenticating user has retweeted Tweet.
12	UCRA:User_Created_at	String	'2006–2009', '2010–2013', . . ., '2022–2025'	UCRA represents the tweet creation time in UTC.
13	UCOO:User_Coordinates	coordinates	YES/NO	This area indicates the geographical location of the tweeting or the user application. The internal coordinates are formatted as geoJSON (longitude first, then latitude). This area can be null.
14	UDPR:User_Default_Profile	Boolean	TRUE/FALSE	When UDPR is set to True, it means the user's profile theme or background remains unchanged.
15	UFAC:User_Favorite_Count	Boolean	'0–9', '10–19', . . . '100,000–1,999,999'	This feature displays the approximate number of times users have expressed their approval of a tweet by selecting the "like" option on the Twitter interface.
16	UFAV:User_Favorited	Int	'0–9', '10–19', . . . '900–999'	This field indicates whether the tweet in question was liked by the authenticating user.
17	URSN:User_in_Reply _to_ScreenName	String	YES/NO	If the Tweet is a reply, the screen name of the original tweet's author will be displayed here.
18	UPSE:User_Possibly_Sensitive	Boolean	TRUE/FALSE	The presence of this field is contingent upon the inclusion of a link within the tweet. Its purpose is not directly about the tweet's content, but rather to indicate that the URL in the tweet may contain sensitive content or media.
19	UPRO:User_Protected	Boolean	TRUE/FALSE	If this is true, it signifies that the user has elected to safeguard their tweets.
20	URTC:User_Retweet_Count	Int	'0–9', '10–19', . . . '100,000–1,999,999'	This dataset contains information about how many times a specified tweet was retweeted.

Table 3. *Cont.*

No	Type in Model	Type	Evaluation Range	Explanation
21	UURL:User_Url	String	YES/NO	The user is prompted to provide a Uniform Resource Locator (URL) in relation to their profile.
22	UVFD:User_Verified	Boolean	TRUE/FALSE	When the value of UVFD is true, it implies that the user has an authenticated account.
23	CLASS:Account Suspender	Boolean	TRUE/FALSE	CLASS represents the class label and indicates whether an account is spam.

The table shows that the dataset comprises 23 distinct features. Column one lists their names and abbreviations, column two specifies their data type classifications, column three outlines their permissible value ranges, and the last column offers descriptive explanations. During dataset structuring, binary features (e.g., YES/NO or TRUE/FALSE) were converted to numerical representations ('1' or '0'), whereas other features, except the UCRA feature, were assigned integer values falling within their designated assessment intervals. The UCRA feature exclusively contains year-based values from a predefined time frame. The complete structural details of Dataset 3 appear in the data encoding and conversion subsection of the data preprocessing phase.

3.2. Light-ANFIS Architecture for Spam Detection in Social Networks

The light-ANFIS architecture represents a novel method that leverages adaptive network-based fuzzy inference systems as an efficient and simplified spam detection solution for social networks. The innovation stems from techniques mirroring the methodologies employed by LightGBM, particularly its speed-optimized data preprocessing prior to classification, and their functional significance within this framework. LightGBM executes a dual-phase procedure during data preprocessing: feature selection and dimension reduction. The GOSS technique minimizes data size by preserving the distribution of samples, whereas Exclusive Feature Bundling (EFB) trims feature size without compromising model accuracy. The Light-ANFIS architecture adopts GOSS, which mirrors the approach of LightGBM for data sample reduction but introduces RFBFC as an innovative alternative for feature reduction. Integrating RFBFC and GOSS into ANFIS's core architecture shrinks dimensionality and decreases data sizes. This dual optimization simplifies spam detection and accelerates the training process.

Figure 2 illustrates the schematic diagram of Light-ANFIS architecture, which is proposed as a high-performance spam detection model for social networks. As depicted in the diagram, this method organizes the spam detection process into two basic phases: data processing and data assessment.

Direct spam detection cannot be performed on the unprocessed data obtained using the Twitter API. The data processing phase prepares the data for subsequent assessment phase. Here, feature extraction and data encoding/conversion procedures are used. Feature extraction involves cleansing irrelevant characters from incoming data, followed by feature extraction and labeling through crowdsourcing. Meanwhile, data encoding/conversion converts categorical features (with bounded value ranges) and binary features into real-numbered representations, which are then stored alongside class labels in the database. The data assessment phase classifies the processed data as spam or non-spam. The execution involves four basic procedures: RFBFC, Data Splitting, GOSS, and ANFIS. RFBFC merges dataset samples via feature-gain-based clustering, enabling dimensional reduction and an architecturally simpler ANFIS configuration. The data splitting procedure divides the

dimensionally reduced data at a ratio of 0.7 to separate it into training and test sets. With the GOSS procedure, an average of about 40% data sample reduction was performed to implement an effective and fast training process for ANFIS. In this process, the test data is applied to the trained ANFIS model to perform spam status detection. All processes related to the data processing and assessment phases are explained in detail in the subheadings below.

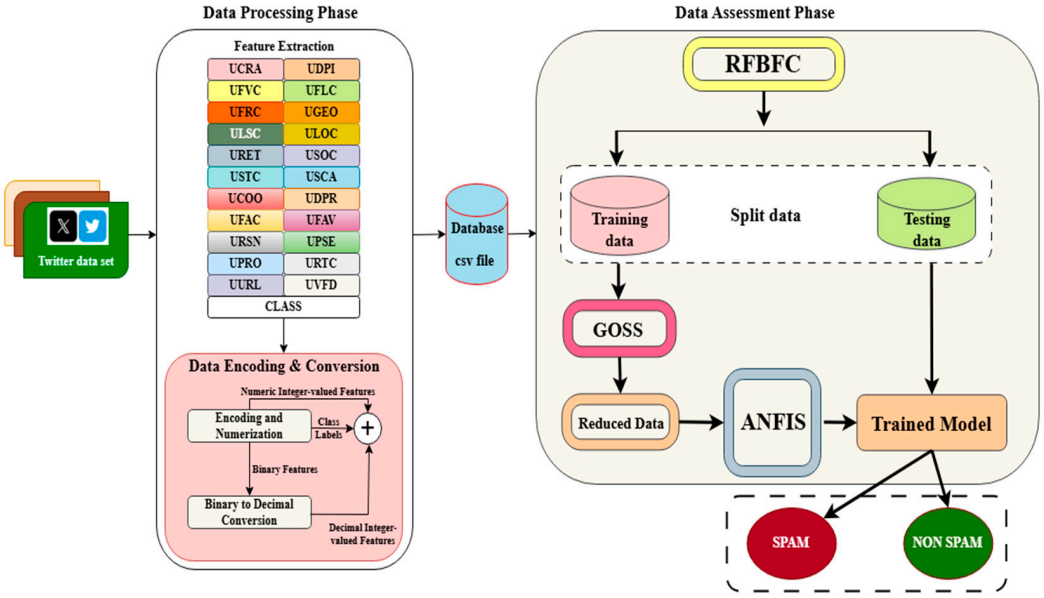


Figure 2. Schematic diagram of the proposed Light-ANFIS architecture for spam detection in social networks.

3.2.1. Data Processing Phase

The data processing phase involves encoding and conversion processes to process raw Twitter data, extract its features, create labels, and prepare the features for the data assessment phase. The following subheadings summarize the operations used to perform these procedures.

Feature Extraction

Python programs retrieve data together with other accompanying features. These raw data in JSON format contain numerous features that may be meaningless, performance-degrading, or potentially harmful to the operation. Before determining which features belong to spam and genuine accounts, it was evaluated whether they should be included in the feature extraction set. These typically manifest as stop words, emojis, punctuation marks, and numbers. This process is called dataset normalization [62]. This method, known for detecting Out-Of-Vocabulary (OOV) words, identified and cleaned meaningless characters in the dataset. Thus, as in natural language processing, various expressions in the language are converted into a single standard, resulting in a more meaningful and cleaner dataset. The crowdsourcing method determines spam in the dataset by extracting raw real classes and features.

The labeling process for Dataset 3 was performed using a crowdsourcing platform. Participants completed the informed consent procedure before starting the study, with only those who voluntarily agreed to undertake the tasks. The participants were remunerated for their contributions in accordance with the standard platform policies. The platform maintains a clear framework concerning ethical standards and participant rights. Our research was conducted in accordance with the ethical guidelines of the host institution.

Data Encoding and Conversion

Feature extraction produced features expressed in many data formats, including "tweets", "arrays of objects", "integer", "string", "coordinates", and "boolean" as well as evaluation intervals defined by starting and ending values with boolean values. Features defined with Boolean values were represented as "YES/NO" and "TRUE/FALSE", and others included features given as interval values like "0–9", "2006–2009", and "100,000–1,999,999". The feature values, which do not directly represent numerical values, must be converted into a format compatible with the Light-ANFIS architecture, specifically, decimal number values. This study used numbers in integer form, although they may appear as integers or decimals. The data encoding and conversion process was executed in two phases. The first phase involved data encoding and numerization. Initially, the features were categorized into boolean and decimal groups based on their non-numeric states. For the boolean group, non-numeric values like "YES/NO" and "TRUE/FALSE" were replaced with "1" for "YES/TRUE" and "0" for "NO/FALSE," encoding all values as binary digits (0 s and 1 s). The decimal group features have values formatted as "minValue-maxValue," where each feature value corresponds to a decimal number within this range. Representing data with multiple values (rather than a single value) within a specified range improves data accuracy. Thus, each data point in this feature range was represented by five randomly generated values. Similarly, the boolean group features representing the same data were replicated five times based on their current value (0 or 1). At the end of encoding and numerization, the extracted data were expanded fivefold in the sample count, yielding three data groups: decimal-valued, binary-valued, and class label. This process is outlined in the pseudocode for the encoding and numerization procedures in Algorithm 2.

Algorithm 2 Coding and Numerization of Features

```
Input     spamDataset          // Original dataset
Outputs   numrFeatures         // Numeric features
          booInFeatures        // Boolean features
          classLabels          // Class labels
Begin
[rows, cols] = size(spamDataset)    // Process all columns including class labels
for col = 1 to cols do
  colNumrFeatures = [], colBoolnFeatures = []
 for row = 1 to rows do
  value = spamDataset[row, col]
// Assign a boolean value to the column and make 5 copies
    if value in ["YES", "NO", "TRUE", "FALSE"] then
        numericBooln = (value IN ["YES", "TRUE"]) ⇐ 1
        ["NO", "FALSE"] ⇐ 0
        boolnArray = [numericBooln, numericBooln, numericBooln,
numericBooln, numericBooln]
         colBoolnFeatures.ADD(boolnArray')
        // Generate and assign 5 numeric values randomly
        // in the related range for the column
    else
        if contains(value, "-") then
        [min, max] = parse_range(value)
        randomNums = []
        for i = 1 to 5 do
```

Algorithm 2 *Cont.*

```
15.                 randomNums.Add(RANDOM_INT(min, max))
16.               end for
17.               colNumrFeatures.Add(randomNums')
18.             end if
19.           end if
20.         end for
21.       numrFeatures.Add(colNumrFeatures)
22.       boolnFeatures.Add(colBoolnFeatures)
23.       end for
24.         // Extract class labels from processed boolean features
          classLabels = boolnFeatures[cols]
25.         // Remove class labels from boolean features
          boolnFeatures.Remove_At(cols)
26.       Return numrFeatures, boolnFeatures, classLabels
27.       end
```

The binary-valued features extracted in the first phase were converted into decimal numbers during the second phase of data encoding and conversion process. The pseudocode in Algorithm 3 illustrates that it consists of multiple steps: determining the significance of binary-valued features based on class labels, sorting ranked features, and finally, converting the sorted binary features into decimal values.

Algorithm 3 Binary to Decimal Conversion of Features

```
      Input     boolnFeatures, classLabels    // Boolean feature set, Class labels
      Outputs   DecFeature                    // Decimal Feature
      Begin
      // Rank features using Random Forest importance
1.    rfmodel = train_random_forest(boolnFeatures, classLabels)
2.    Feature_importances = get_feature_importance(rfmodel)
3.    [~, rankedIndices] = sort(Feature_importances, 'descend')
4.    rankedFeatures = boolnFeatures(:, rankedIndices)
       // Convert each row from binary to decimal
5.    decValues = zeros(size(boolnFeatures, 1), 1)
6.    for i = 1:size(boolnFeatures, 1)
7.      binaryrow = rankedFeatures(i, :)
8.      powers = 2.^(length(binaryrow)-1:-1:0)
9.      decValues(i) = sum(binaryrow .* powers)
10.   end for
11.   DecFeature = decValues
12.  end
```

This study employs a RF-based feature selection method for both converting binary features to decimal and ranking/ordering features within our proposed RFBFC technique. The method is advantageous because it (a) efficiently ranks features across different data formats, (b) performs feature ranking rapidly, and (c) effectively captures nonlinear relationships between features [63]. Using the RF algorithm, binary features are ranked by the Gini index and reordered in descending sequence, forming a matrix where each row

represents a binary number. The binary-to-decimal conversion formula in Equation (2) is then applied row-wise to convert these binary feature columns into a single integer column.

$$binaryrow_i = [A_{11}\ A_{12}\ A_{13} \ldots\ldots\ A_{1n-1}\ A_{1n}] \qquad \text{j} = 1, 2, \ldots \text{n} \tag{1}$$

$$DecFeature_i = 2^{n-1} \times A_{11} + 2^{n-2} \times A_{12} + 2^{n-3} \times A_{13} + \ldots\ldots + 2^1 \times A_{1n-1} + 2^0 \times A_{1n} \tag{2}$$

Here, $binaryrow_i$ is the binary number sorted by the RF-based feature selection method in the *i*-th row, $DecFeature_i$ is the decimal equivalent of the binary number in the i-th row, A_{1j} is the sorted feature in the 1-st row and j-th column, and $\left[2^{n-1}\ 2^{n-2}\ 2^{n-3} \ldots\ldots\ 2^1\ 2^0\right]$ is the digit weight array used for binary-to-decimal conversion. 2^{n-1} represents the weight of the n-th digit, while 2^0 represents the weight of the first digit. Following this process, the numerical integer-valued feature matrix (*numrFeatures*) derived from data encoding and numerization, the integer-valued feature matrix (*DecFeature*) obtained by binary-to-decimal conversion, and the class label column matrix (*classLabels*) were merged to form the dataset for the data assessment phase.

Feature Importance Analysis

Feature importance analysis is a crucial component for AI studies that offers several essential functions. This study improves the accuracy and reliability of the model while ensuring its interpretability. In particular, in problems with high false-positive/negative risks, such as spam detection, an explanation of the features that inform classification decisions enhances the method's reliability and enables the proposed model to be implemented in practical applications. After completing the preprocessing in this subsection, RF-based feature analysis was performed for all three datasets. Figure 3 illustrates the results of the feature importance scores derived from the RF feature selection experiments conducted on Dataset 1, Dataset 2, and Dataset 3.

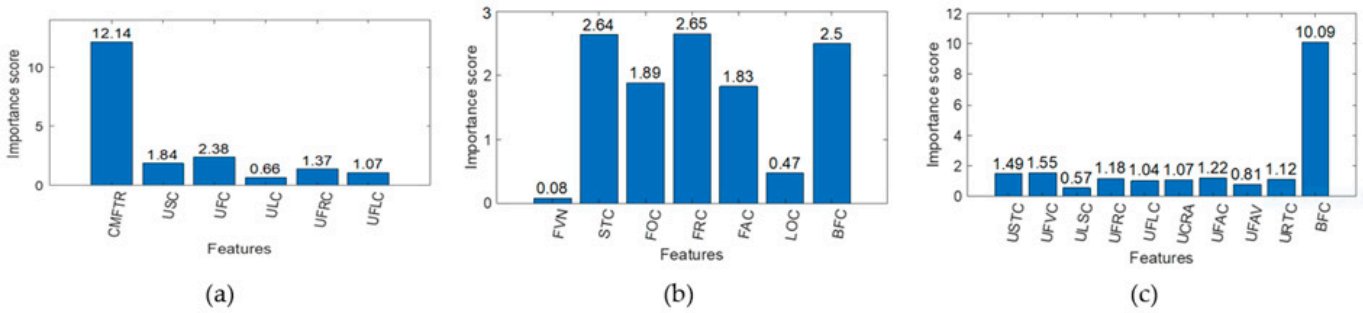


Figure 3. Overview of feature importance for Dataset 1, Dataset 2, and Dataset 3: (**a**) Dataset 1, (**b**) Dataset 2, and (**c**) Dataset 3.

From the visual feature importance analysis for Dataset 1 in Figure 3a, it is understood that the CMFTR feature accounts for a score of 12.14 out of a total importance score of 19.46. The remaining scores were assigned to the USC, UFC, ULC, UFRC, and UFLC features, providing scores of 1.84, 2.38, 0.66, 1.37, and 1.07, respectively. According to the raw scores, the relative importance degree of each feature was calculated as follows: CMFTR, USC, UFC, ULC, UFRC, and UFLC scores were 62.38%, 9.46%, 12.23%, 3.39%, 7.04%, and 5.50%, respectively. The high importance degree of 62.38% among all features for the CMFTR feature makes it dominant in determining the classification score compared with other features. The considerable dominance of CMFTR arises from its nature as a composite feature, which is the representation of binary features ranked by class label in decimal conversion. In other words, the score generated by the CMFTR indicates the total importance score of the six binary features. These findings indicate that the impact of

the binary features comprising Dataset 1 on determining the classification result (62.38%) is higher than that of the real-valued features (37.62%). In the visual feature importance analysis for Dataset 2 in Figure 3b, a more balanced distribution in terms of importance degrees compared to Dataset 1 was achieved. Importance scores along with grades of 0.08 (0.66%), 2.64 (21.89%), 1.89 (15.67%), 2.65 (21.98%), 1.83 (15.17%), 0.47 (3.90%), and 2.5 (20.73%) were obtained for the FVN, STC, FOC, FRC, FAC, LOC, and BFC features Where BFC indicated the decimal-valued combined binary features. Considering the BFC score of 20.73%, it can be inferred that the total impact of binary-valued features on the classification score for Dataset 2 is lower than that of the real-valued features, which include FVN, STC, FOC, FRC, FAC, and LOC. Individually, STC and FRC exhibited the most significant impact on classification, whereas FVN and LOC showed the smallest effect. The visual feature importance analysis results for Dataset 3 in Figure 3c are similar to those of Dataset 1, except for the feature redundancy. The scores and computed importance degrees for the features of USTC, UFVC, ULSC, UFRC, UFLC, UCRA, UFAC, UFAV, URTC, and BFC were 1.49 (7.40%), 1.55 (7.70%), 0.57 (2.83%), 1.18 (5.86%), 1.04 (5.16%), 1.07 (5.31%), 1.22 (6.06%), 0.81 (4.02%), 1.12 (5.56%), and 10.09 (50.10%), respectively. The BFC feature illustrates decimal-valued, combined binary features. The remaining features were real-valued. Similarly to Dataset 1, the BFC, which reflects the total impact of binary features, achieved a notably high importance degree, reaching as high as 50.10%. The total effect of real-valued features, which had a minimal impact, was approximately 49.9%.

For Dataset 1 and Dataset 3, the results of the feature importance analysis showed that decimal-valued combined binary features were dominant characteristics with significantly high importance scores. Under these conditions, a classification process may yield results in which dominant features determine the classification result, while other features become ineffective. Due to the cluster determination principle of the RFBFC method based on the feature score standard deviation, high standard deviation ratios cause dominant features to be placed in one cluster, while low-scoring features are assigned to another cluster. Combining these low-scoring features in the same cluster results in a new composite feature that reveals the total effect of existing real features. This reduced feature structure eliminates the dominant feature effect and creates a new feature set. The RFBFC method obtains the feature reduction process for Dataset 2, which has a more balanced feature distribution than both datasets, by combining features that fall into clusters determined by the standard deviation ratio formed according to the feature scores. The reduced feature count varies according to the feature distribution of each cluster.

3.2.2. Data Assessment Phase

The data assessment phase involves processing and classifying data transferred to the database after the completion of the data processing phase to determine whether it is spam. During this phase, pre-processed data undergo sequential steps, including RFBFC, data splitting, and GOSS, before being classified using the ANFIS architecture. This subsection clarifies the components of the techniques and methodologies relevant to the processes involved.

Random Forest-Based Feature Clustering (RFBFC) Technique

The RFBFC technique involves clustering features based on gain-based columns after feature gain ranking and then merging them within the same cluster. This approach aims to achieve efficient feature reduction through a three-phase process: “Ranking,” “Clustering,” and “Merging.” During the feature ranking process, RF, a method proven effective for such problems in recent years, was used [64]. RF is an advanced ML algorithm that relies on the average prediction of decision trees generated during training. It efficiently selects features

by incorporating randomness in feature selection and employs the bagging (bootstrap aggregating) technique to process training datasets [65]. In fact, feature selection in this method occurs naturally because trees are split at each training step. Each tree is constructed using randomly sampled training data, while some samples, called out-of-bag (*OOB*), are excluded. Although *OOB* samples are not directly used in training, their OOB error is a key indicator of the model's generalization performance. The input parameters derived from the *OOB* samples provided additional insights. When the number of trees is low, feature importance can be easily assessed under specific criteria; however, determining importance becomes more challenging as the tree count exceeds 50 [66,67]. The *OOB* Permuted Predictor Delta Error method of Matlab 2021a enables effective feature ranking in both scenarios. Thus, our proposed RFBFC technique employs this approach for the feature evaluation. Based on Breiman's permutation importance [68], the *OOB* Permuted Predictor Delta Error method measures the rise in prediction error when each feature is randomly permuted in the *OOB* samples of a trained RF model. Consider a random forest with an $N \times P$-dimensional feature matrix X, target vector Y, and K trees. For feature ranking, the *OOB* samples of each tree k (OOB_k) are identified, and the tree is trained using a bootstrap sample. The original error for these samples was then computed using the formula provided in Equation (3).

$$error^{k}_{orginal} = \frac{1}{|OOB_k|} \sum_{i \in OOB_k} (y_i - \hat{y}_i)^2 \tag{3}$$

Here, $error^{k}_{orginal}$ indicates the *OOB* error of the k tree, $|OOB_k|$ indicates the *OOB* sample count of the k tree, $\hat{y}_i$ indicates the prediction of the k tree for the i-th observation, and y_i indicates the target of the k tree for the i-th observation. Then, the permutation measurement, a three-step process, was commenced. In the first process step, *OOB* samples were randomly permuted for each x_j feature (j-th column). Then the i-th observation prediction of the k-th tree for this feature was computed.

$$\hat{y}^{k,j}_{i,perm} = Tree_k\left(x^{j}_{i,perm}\right) \tag{4}$$

In the third process step, the *OOB* permutation error is calculated via Equation (5) to evaluate changes in model performance.

$$error^{k,j}_{perm} = \frac{1}{|OOB_k|} \sum_{i \in OOB_k} \left(y_i - \hat{y}^{k,j}_{i,perm}\right)^2 \tag{5}$$

Afterwards, for the k-tree used to assess feature importance, the delta error of the j-th feature ($\Delta error^{k,j}$) is computed by the difference between the *OOB* original error and the permutation error. The higher the error value for a feature, the greater its importance. The gain-based importance of each feature can be determined by averaging the delta errors across trees.

$$G_J = \frac{1}{K} \sum_{k=1}^{K} \Delta error^{k,j} \tag{6}$$

$$\Delta error^{k,j} = error^{k,j}_{perm} - error^{k}_{orginal}$$

G_J specifies the gain value of the j-th feature. Gains are scaled in the range of 0–1 to increase interpretability and computational efficiency. For this purpose, the normalized gain vector (G^{norm}_j) was obtained by applying min-max normalization to the M-dimensional gain vector.

In the clustering phase, the standard deviation of the normalized gain vector is calculated as the first step. The basic formula for this step is presented in Equation (7).

$$\sigma = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(G_j^{norm} - \overline{G^{norm}}\right)^2} \quad (7)$$

In Equation (7), M is the number of features and $\overline{G^{norm}}$ is the average of the normalized gain vector. The subsequent step was to obtain the interval vector for clustering. This vector is formed as a descending numerical sequence from 1 to 0, decremented by σ at each step ($1, 1-\sigma, 1-2\times\sigma, \ldots, 1-\mathrm{N}\times\sigma, 0$). The sequence, which contains N + 2 elements, defines the boundaries of the potential feature clusters, with each consecutive pair marking the cluster limits. Here, N + 1 clusters exist, and since $1-\mathrm{N}\times\sigma$ must be greater than 0, N is determined as the largest integer less than $1/\sigma$. Once the cluster boundaries are set, features are assigned to the clusters based on their gain values. Any empty clusters were removed, finalizing the clustering process. In the third and final phase, the features within the same cluster were combined by applying the weighted average method, taking into account their gains and their own values. The formulation of the combined feature defining this process for the l-th cluster with K features obtained in the clustering process is given in Equation (8).

$$CMBF_l = \frac{\sum_{k=1}^{K} G_k^{norm,\,l} \times X_k^l}{\sum_{k=1}^{K} G_k^{norm,\,l}} \quad (8)$$

Here $CMBF_l$ is the combined feature value for the l-th cluster, $G_k^{norm,\,l}$ is the normalized gain of the k-th feature in the cluster, and X_k^l is the value of the k-th feature in the cluster l.

Now, let us explore what differentiates RFBFC from dimensionality reduction techniques such as Exclusive Feature Bundling (EFB), feature selection, and Principal Component Analysis (PCA). To better understand RFBFC's unique advantages over these methods, we will briefly examine these techniques and their limitations. EFB is an efficient LightGBM component that bundles mutually exclusive features into a single feature bundle [69]. It operates effectively when exclusive features are present. However, in spam texts, the frequent co-occurrence of certain words may lead to incorrect grouping by EFB. Merging features in an exclusive form can result in interpretability differences owing to the potential loss of meaningful features. Although EFB achieves effective feature compression, it fails to capture complex word relationships. Feature selection techniques such as χ^2, MI, and LASSO have been successfully applied to classification problems, especially with ML methods, owing to their ability to capture linear or nonlinear relationships [70]. However, significant challenges arise when applying these techniques to fuzzy inference systems (FIS), such as ANFIS. Maintaining a limited number of input features is crucial for the functional ANFIS configuration. Consequently, excluding certain effective features in spam detection leads to issues such as the inability to learn nonlinear connections and to capture contextual relationships. These problems become more pronounced when dealing with a large number of features. On the other hand, PCA is a dimensionality reduction technique widely used in ML methods [71], which extracts eigenvalues and eigenvectors from the covariance matrix to create principal components that explain variance. However, its reliance on a linear variance structure makes it challenging to capture semantic relationships in nonlinear structures, such as spam data. Furthermore, its focus on maximizing variance may result in the exclusion of even the most critical data features. Dimensionality reduction using PCA, even at moderate levels, may fail to produce acceptable inputs for FIS methods such as ANFIS.

The RFBFC method contributes significantly to creating effective inputs in the ANFIS architecture and forming a simplified ANFIS configuration by solving the major limitations of the aforementioned dimensionality reduction techniques through RF-based feature ranking, clustering using gain standard deviation, and weighted feature combination. Employing RF within the method allows obtaining reliable importance degrees that are not solely based on variance or correlation but are directly related to the target because RF is a highly effective method for capturing nonlinear relationships and learning complex interactions between features and the target variable. The RFBFC method's shaping of the clustering process through standard deviation provides several practical advantages. The standard deviation measures the natural spread of the distribution. This allows similar characteristics (homogeneous or heterogeneous) to be grouped together. In other words, features that are close to each other in a random direction, such as low, medium, or high, are represented in the same cluster. Consequently, the entire dataset can be represented without feature loss. Setting boundaries starting from 1 and decreasing by the amount of standard deviation enables the clusters to be determined in a data-specific and flexible manner. Combining features based on their weights within the same cluster allows the method to maintain data integrity during dimensionality reduction better. This structure allows the RFBFC method to address the key limitations of the above dimensionality reduction techniques. Evaluating all features within their own category prevents the elimination of low-rank features by feature selection methods, thereby avoiding the partial information loss that would otherwise occur. RF feature rating establishes a relationship with the target variable, while clustering and weighting eliminate the interpretability challenges of feature significance found in PCA, making it easier to interpret. Furthermore, it overcomes the limitations inherent in EFB through effective clustering and weighted combinations.

Gradient-Based One-Sided Sampling (GOSS) Technique

The proposed method employs the GOSS technique for rapid and efficient training without compromising performance. This innovative approach performs down sampling according to the sample gradients in a dataset [72]. GOSS operates on the principle that well-trained samples have small gradients, while under-trained samples have large gradients; thus, it randomly samples the former while retaining the latter. When implementing the algorithm of this technique, the absolute gradients of the samples were first calculated and sorted by value. Samples with large gradients are then selected at a rate of $a \times 100\%$ and preserved. Small gradient samples are selected from the remaining samples at a rate of $b \times 100\%$ and weighted by the coefficient $(1 - a)/b$ to create a balanced data distribution [69,72,73]. This technique combines the selected large and small gradient samples to produce the final result set. Algorithm 4 illustrates this process with pseudo-code given below.

The pseudocode in Algorithm 4 outlines the following steps: Step 1 scales the general distribution of the small-gradient-sample dataset by 1/b. Step 4 provides the preliminary prediction results. Step 5 computes the gradient gains. Step 6 sorts the results obtained. Step 7 selects the large-gradient samples. Step 8 randomly selects a small gradient sample from the remaining samples. Step 9 merges the two selected data segments. Step 10 assigns weights to the small-gradient data. Finally, Step 11 constructs a new dataset.

Algorithm 4 GOSS Technique

Input(s) I: training data, d: iterations
a: sampling ratio of large gradient data,
b: sampling ratio of small gradient data
loss: loss function, L: weak learner

Steps

```
models ← {}, fact ← (1 − a) / b
topN ← a × len(I), randN ← b × len(I)
for i = 1 to d do
    preds ← models.predict(I)
    g ← loss(I, preds), w ← {1,1, . . .}
    sorted ← GetSortedIndices(abs(g))
    topSet ← sorted[1: topN]
    randSet ← RandomPick(sorted[topN: len(I)], randN)
    usedSet ← topSet + randSet
    w[randSet] = fact × w[randSet]
    newModel ← L(I[usedSet], − g[usedSet], w[usedSet])
    models.append(newModel)
end for
```

Adaptive Neural Fuzzy Inference System (ANFIS) Architecture

The model development and testing experiments were performed using a program written in Matlab. Consequently, this study employed an ANFIS architecture supported by the Matlab programming environment and functions. This system incorporates a first-order Sugeno fuzzy inference system, which is characterized as a neural network structure. Integrating neural networks into the fuzzy inference system enables parameter optimization based on fuzzy sets and rules, thereby reducing potential expert-related issues. Structurally, ANFIS consists of functional layers and interconnecting links, similar to neural networks [74,75]. However, unlike standard neural networks, their layers function as components of a Sugeno-type inference system.

In the proposed Light-ANFIS architecture, the inputs to ANFIS were obtained using the RFBFC technique as combined features. Therefore, the number of inputs to this unit is equal to the cluster count created by the RFBFC technique based on the incoming features. Figure 4 shows a schematic of the ANFIS with two inputs, one output, and 25 rules determined by the RFBFC technique according to the nature of the features. The rule count in the ANFIS architecture depends on the fuzzy set count assigned to each input. This was determined through ablation studies, the results of which are presented in Section 4. These studies revealed that the optimal configuration, which balances simplicity and performance, employs Gaussian membership functions with five sets per input. Thus, the ANFIS configuration adopted five membership sets for each input variable.

As shown in Figure 4, this architecture comprises a five-layer structure with processing nodes, where each node executes a specific function in the fuzzy inference system.

Layer 1 (Fuzzification): This layer receives features merged via the RFBFC technique from the input node and applies fuzzification to them. This process involves verbally labeling incoming inputs and computing the membership degrees for each label. It operates through nodes, where each input is represented by five fuzzy sets with Gaussian membership functions. The output of each processing node in this layer is derived using the formula in Equation (9), as follows:

$$O_{ij}^{1} = \mu_{CMBF_{ij}}(CMBF_i) \;\; i = 1,2 \qquad j = 1,2,3,4,5 \tag{9}$$

$$\mu_{CMBF_{ij}}(CMBF_i) = e^{-\frac{(CMBF_i - c_j)^2}{2\sigma_j^2}}$$

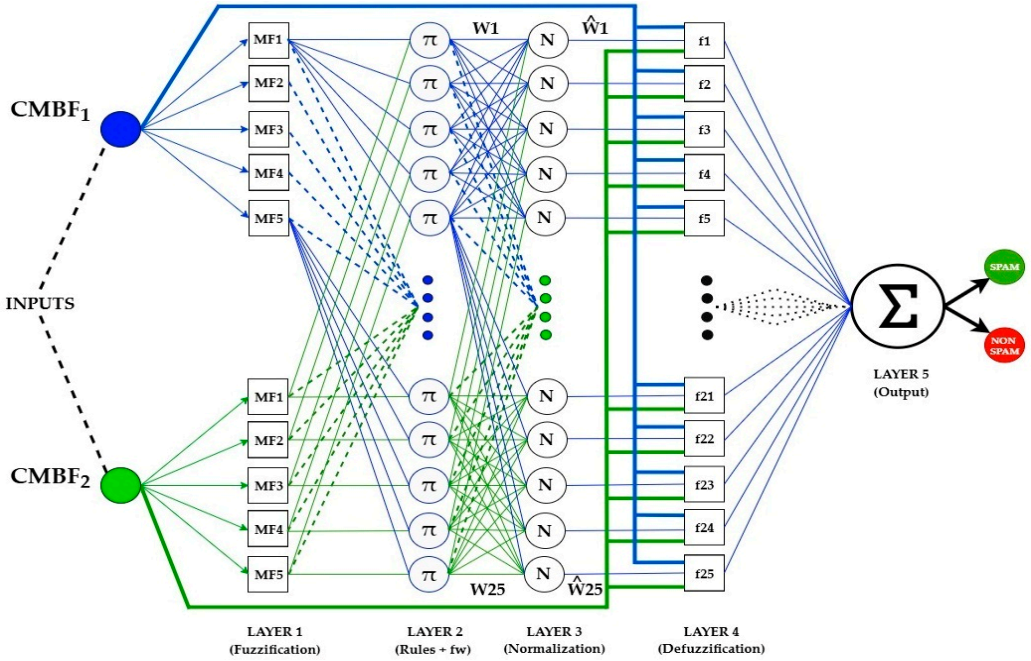


Figure 4. Schematic diagram of the ANFIS with two inputs, one output, and 25 rules.

Here, O_{ij}^1 indicates the output node for the j-th set of the i-th input in Layer 1, $CMBF_i$ represents the i-th combined feature input for ANFIS, $\mu_{CMBF_{ij}}$ defines the membership function for the j-th set of the i-th input, and (c, σ) denotes the antecedent parameters of the Gaussian membership function.

Layer 2 (Rules & fw): By applying the T-norm operator to the membership degrees of the inputs from Layer 1, the firing weight is determined for each rule. The T-norm operator employs product and minimum operations. In Sugeno-type ANFIS configurations, the product operator is commonly used, and the firing weight for the l-th rule (Layer 2, l-th node) can be computed using the formula provided in Equation (10), as follows:

$$O_l^2 = w_l = \mu_{CMBF_{1j}}(CMBF_1)\mu_{CMBF_{2k}}(CMBF_2) \quad \begin{matrix} j = 1,2,\ldots,5 \quad k = 1,2,\ldots,5 \\ l = 1,2,3,4, \quad \ldots\ldots 24, \ 25 \end{matrix} \tag{10}$$

Here, O_l^2 is the l-th node output in Layer 2, w_l is the firing weight for the l-th rule, j is the membership set index for the first input, k is the membership set index for the second input, and l is the rule index.

Layer 3 (Normalization): In this layer, known as the normalization layer, the node outputs from Layer 2 are normalized relative to the total firing weight, producing normalized fired rule weights ($\hat{w}_l$).

$$O_l^3 = \hat{w}_l = \frac{w_i}{\sum_{k=1}^{25} w_l} \tag{11}$$

Layer 4 (Defuzzification): This layer determines the contribution of each rule to the final outcome. It processes the rules to compute output values. The output of every node in Layer 4 can be determined using Equation (12).

$$O_l^4 = \hat{w}_l \times f_l \tag{12}$$

$$f_l = (p_l \times CMBF_1 + q_l \times CMBF_2 + r_{kl})$$

where (p_l, q_l, r_{kl}) represents the consequent parameters of the l' th rule.

Layer 5 (Output): This layer, which is composed of a single summation node, produces the primary output of the ANFIS architecture by aggregating the outputs from Layer 4. The mathematical expression for this output is as follows:

$$O^5 = \sum_{l=1}^{25} \hat{w}_l \times f_l = \frac{\sum_{l=1}^{25} w_l \times f_l}{\sum_{l=1}^{25} w_l} \quad (13)$$

$\hat{w}_l \times f_l$ corresponds to the output of the l-th node in Layer 4.

ANFIS training employs backpropagation and hybrid-learning algorithms. Although backpropagation is straightforward to implement, it often lags behind hybrid learning in terms of efficiency and speed. Hence, this study uses a hybrid learning algorithm to train the ANFIS. During ANFIS learning, the antecedent and consequent parameters of the rule structure were optimized. This algorithm integrates the least squares (LSE) method with gradient descent (GD), updating the consequent parameters via LSE and the antecedent parameters via GD [76,77]. Training ends when the error between the actual and predicted values meets the desired threshold or the predefined number of epochs is reached.

3.3. Performance Assessment

Confusion matrix metrics are commonly used in scientific studies, ranging from classification to segmentation, to assess the effectiveness of proposed models in addressing specific problems [78–80]. This study introduces the Light-ANFIS architecture, a binary classification model for spam detection in social networks. Compared to the traditional ANFIS architecture, Light-ANFIS offers a simpler configuration and faster training. The model performance evaluation incorporates not only confusion matrix metrics but also training time and training error (RMSE) measurements. Table 4 presents the relevant formulas and fundamental definitions of these metrics [81].

The confusion matrix is a simple table that evaluates a model's predictions against the actual labeled results, showing correct and incorrect classifications. In binary classification, this matrix operates using four key parameters: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP counts correctly identified positive samples, FP counts negative samples incorrectly labeled as positive, TN counts correctly identified negative samples, and FN counts positive samples incorrectly labeled as negative. The matrix is the foundation for calculating five performance metrics derived from the TP, TN, FP, and FN values: accuracy, recall, F-score, precision, and AUC [82]. Among these metrics, accuracy indicates the overall performance of the model. Although experiments on this metric using balanced datasets provide successful results, they may produce misleading results with unbalanced datasets. In cases where identifying positive prediction classes are critical, recall and precision metrics are used. The F-score is a metric that establishes a balance between recall and precision and plays a significant role in accurately measuring the performance of imbalanced datasets [82–84]. The AUC metric evaluates the model's capability to differentiate between the positive and negative classes. Ranging from 0 to 1, a result closer to 1 indicates near-perfect class separation. The RMSE and training time metrics were used to evaluate the training performance of the model. The RMSE quantifies the average prediction error relative to the actual values during training. As this value approaches 0, the correct prediction performance of the model improves. As a training process, the development of fast models emerges as an indicator in terms of their usability in real-time applications and large datasets. Because our model is designed for this purpose, the training time metric plays a vital role in evaluating the training process performance.

Table 4. Performance metrics, formulas, and definitions used in the evaluation of Light-ANFIS architecture.

Metrics	Formula	Definition
Accuracy (Acc)	$\text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$	This metric calculates the proportion of samples that were correctly classified from all samples that were evaluated.
Recall (Rec)	$\text{Rec} = \frac{TP}{TP+FN}$	Recall is an indicator of how accurately the model measures positive samples.
Precision (Prec)	$\text{Prec} = \frac{TP}{TP+FP}$	Precision is a metric that gives the ratio between the number of positive samples correctly predicted among the samples that the model identified as positive.
F1-score (F1-scr)	$\text{F1}-\text{Scr} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$	The F1 Score, or F-measure, expresses the harmonic mean of precision and recall. This metric is a valuable indicator when precision and recall are equally important. F1-score (hereafter referred to as F-score).
Area Under ROC Curve (AUC)	$AUC = \int_0^1 TPR(FPR)dFPR$	The Area Under the Curve (AUC) performs the model performance measurement based on the area under the Receiver Operating Characteristic (ROC) curve. In the AUC equation, TPR defines the proportion of correctly classified positive samples, whereas FPR indicates the proportion of incorrectly classified positive samples.
RMSE	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	RMSE is a metric that provides the square root of the mean squared differences between the predicted values and real ones. In this context, y_i represents the i-th predicted value, $\hat{y}_i$ is the i-th actual value, and n is the total number of samples.
Training Time T(t)	–	This represents the time allocated for model training. In practice, it serves as a performance metric that determines the processing and memory complexities of the model.

4. Experimental Results and Discussion

This section presents the ablation studies conducted to develop the proposed Light-ANFIS architecture, the experimental studies performed to demonstrate the effectiveness of this architecture in spam detection on social networks, and an evaluation of the results obtained from them. In addition, the results obtained from the Light-ANFIS architecture were compared with the performances of the current methodologies based on the literature. The ablation and experimental studies were performed on a computer with a 12-th Gen Intel Core i7-12700KF processor running at 3.6 GHz, 64 GB of RAM, and an NVIDIA GeForce RTX 3070 Ti GPU operating on Windows 11. The experiments involving the Light-ANFIS and RFBFC-ANFIS architectures applied to Dataset 1, Dataset 2, and Dataset 3 were executed using a program written in Matlab 2021a. Matlab contains ready-made functions for fuzzy inference-based systems, which reduces the need for coding and includes flexible and wide-ranging tool components for the design process, making it a preferred platform for writing the code to be used in the implementation of the proposed application. In experiments conducted under the 0.7 split condition, the parameter values for the GOSS and RFBFC techniques in the Light-ANFIS and RFBFC-ANFIS architectures were used as specified in the respective section headings, while the ANFIS architectures employed the values listed in Table 5.

Table 5. Setting parameters used in ANFIS architectures and their assigned values.

Parameter' Name	Parameter' Value
EpochNumber	200–4000
InitialStepSize	0.005
StepSizeDecreaseRate	0.9
StepSizeIncreaseRate	1.1
DisplayANFISInformation	0
DisplayErrorValues	0
OptimizationMethod	1

In the experimental process, since the focus was on the final results, the parameters "DisplayANFISInformation" and "DisplayErrorValues" were set to "0" to suppress the display of ANFIS details and error values at each epoch. Because the hybrid learning algorithm was employed for parameter optimization in the ANFIS training phase, the "OptimizationMethod" parameter was set to "1". Regarding the step size, there are three main parameters: "InitialStepSize," "StepSizeDecreaseRate," and "StepSizeIncreaseRate." These parameters critically influence ANFIS's training speed and accuracy. Balanced values were assigned to these parameters to ensure robust ANFIS performance. The initial learning rate ("InitialStepSize") was set to a small value like 0.005, to avoid surpassing the opt imal values. A moderate change rate of 10% was considered to avoid instability caused by increases or decreases in error variations. In this context, a value of 0.9 was assigned to the "StepSizeDecreaseRate" parameter as the error reduction factor, while a value of 1.1 is assigned to the "StepSizeIncreaseRate" parameter as the error increase factor. To eliminate potential convergence problems that may arise due to the small step size in experimental studies, the epoch value was determined to be 4000.

4.1. Ablation Studies

A two-stage ablation study was performed to develop the Light-ANFIS method. In the initial stage of these experiments, we focused on determining the optimal number and type of sets per input to create a simpler yet more efficient ANFIS configuration within the proposed framework. In other words, this stage identified a structurally simple, high-performance ANFIS configuration. To accomplish this, the RFBFC technique was first applied to Dataset 1, yielding a feature-reduced dataset. A split ratio of 0.7 was then used to generate the training and test datasets. Subsequently, ANFIS architectures with varying the number of membership set and types were trained using the reduced dataset. At the end of the ablation studies, where the set-based ANFIS configuration was determined, test data were applied to the trained ANFIS architectures, and the performance results were obtained for architectures with different configurations. Table 6 presents the performance results for the ANFIS architectures with different configurations in the first stage of the ablation studies.

In the experiments, three, five, and seven fuzzy sets were used for each input of ANFIS, configured with linear membership functions ("trimf" and "trapmf") as well as bell and gaussian-shaped membership functions ("gbellmf," "gaussmf," and "gauss2mf"). The complexity of the ANFIS architecture increases with the number of parameters, which depends on the input count, membership function parameter count, and rule count. As shown in the last column of the table, for Dataset 1, a two-input ANFIS configuration using the RFBFC technique resulted in 39 parameters with three membership sets, 95 parameters with five sets, and 175 parameters with seven sets per input. The table provides the training times of the ANFIS architectures constructed based on the number and type of input membership sets, located just to the left of these parameters. These durations

clearly indicate that, irrespective of the set type of ANFIS architecture, increasing the parameter count leads to a significant rise in the training time. The remaining columns present the performance metrics of the architecture derived from the confusion matrix. Analyzing the results by membership set count over the membership set type revealed that ANFIS architectures with seven input membership sets exhibited the best performance in the confusion matrix metrics. The highest performance was achieved with an ANFIS architecture with seven membership sets using the gbell membership function, with an accuracy of 0.99238, precision of 0.98743, recall of 1.00000, F1-score of 0.99368, and AUC of 0.99050. However, the 175-parameter ANFIS architecture required an extensive training time of 834.76 s for Dataset 1. This situation will be much longer, especially for large datasets. Thus, in creating the ANFIS configuration, we prioritized a architecture that balances fewer parameters with relatively high performance. In this context, when the results in the table are evaluated, the 5-set, 95-parameter ANFIS architecture with gaussian membership functions performed nearly as well as the best-performing 7-set, 175-parameter ANFIS architecture with gbell membership functions, which achieved an accuracy of 0.99020, precision of 0.98827, recall of 0.99545, F1-score of 0.99185, and AUC of 0.98891. The former architecture was trained in only 166.97 s, which is far faster than the latter's 834.76 s training time. Consequently, the 5-gaussian set ANFIS architecture was selected as the optimal model, excelling in both the training time and performance.

Table 6. Performance results for the ANFIS architectures with different configurations in the first stage of the ablation studies.

MF Type	Number of Set	Accuracy	Precision	Recall	F1 Score	AUC	Training Time (s)	Number of Parameter
Gbellmf	3	0.97115	0.96368	0.98909	0.97622	0.96673	113.86	39
	5	0.98585	0.98818	0.98818	0.98818	0.98527	364.14	95
	7	0.99238	0.98743	1.00000	0.99368	0.99050	834.76	175
Trimf	3	0.94829	0.95974	0.95364	0.95668	0.94697	51.63	39
	5	0.96407	0.96161	0.97909	0.97027	0.96037	162.04	95
	7	0.97768	0.97319	0.99000	0.98152	0.97465	764.53	175
Gaussmf	3	0.97496	0.97563	0.98273	0.97917	0.97305	51.95	39
	5	0.99020	0.98827	0.99545	0.99185	0.98891	166.97	95
	7	0.99183	0.98742	0.99909	0.99322	0.99005	781.79	175
Gauss2mf	3	0.96788	0.96850	0.97818	0.97332	0.96535	65.57	39
	5	0.98040	0.98188	0.98545	0.98367	0.97916	185.22	95
	7	0.98911	0.98736	0.99455	0.99094	0.98777	840.91	175
Trapmf	3	0.93794	0.95313	0.94273	0.94790	0.93676	58.18	39
	5	0.96952	0.95951	0.99091	0.97496	0.96425	172.51	95
	7	0.98258	0.98195	0.98909	0.98551	0.98098	781.22	175

In the second stage of the ablation studies, experiments were performed to specify the optimal parameter values for the GOSS technique, which was applied to further reduce the training time of the architecture for the given datasets. The preservation rate for high-gradient samples (a) and the random sampling rate for low-gradient samples (b) are the key parameters influencing GOSS performance. As can be understood from the equation $a + b \leq 1$, the sum of these parameter values cannot exceed 1. A value of 1 indicated no data reduction. Because our goal is also data reduction without significantly affecting performance, we conducted experiments by considering values of the total gradient ratio in the range of 0.7–0.1. In the first step of the two-step process, seven experiments with varying a and b values were conducted for cases where a + b equaled 0.7, 0.6, 0.5, 0.4, 0.3,

0.2, and 0.1. The two a-b combinations that provided the best performance in terms of accuracy and training time were selected from each sum, as shown in Table 7. From these experiments, performance scores ranging between 0.99020 and 0.94339 were achieved for different a-b combinations in terms of accuracy, respectively. When the sum of the gradient ratios fell to 0.3 or lower, the accuracy decreased below 0.97000. Given this substantial deviation from the peak performance of 0.99020, the process was executed using gradient ratio sums between 0.7 and 0.4. Table 7 shows the best performance results obtained in the first step of the second stage for the gradient parameters.

Table 7. Best performance results obtained in the first step of the second stage for the gradient parameters.

GOSS Parameters			Performance Metrics	
a	**b**	**Sum of Gradian Rates**	**Accuracy**	**Training Time (s)**
0.10	0.60	0.70	0.98476	204.15
0.30	0.40		0.99020	133.86
0.10	0.50	0.60	0.98748	102.24
0.40	0.20		0.98204	101.02
0.30	0.20	0.50	0.98204	83.76
0.40	0.10		0.98040	83.45
0.20	0.20	0.40	0.97714	65.97
0.30	0.10		0.98040	66.32

For Dataset 1, the highest performance was achieved with an accuracy of 0.99020, equal to the performance before applying the GOSS technique, in the a-b combination of 0.30–0.40 with a total gradient ratio of 0.70. The training time dropped from 166.97 s to 133.86 s for this combination. At a total gradient ratio of 0.60, the best a-b combination was obtained with the pair 0.10–0.50, reaching an accuracy rate of 0.98748. For the remaining total 0.50 and 0.40 gradient ratios, pairs 0.30–20 and 0.30–0.10 delivered-highest accuracy. When the results in the table are examined in general, as the sum value of gradient ratios increases, the performance in terms of accuracy improves slightly. However, increasing the total gradient ratios also led to a longer training time for the proposed architecture.

In the second step of the second stage, a common gradient pair applicable to all three datasets used in the study was identified using the a and b gradient pairs that yielded the highest accuracy performance in experiments with Dataset 1 from Table 7. In this context, to determine an effective gradient pair throughout parameter changes, 12 experiments were conducted on all three datasets, where these pairs were tested in terms of accuracy, training time, recall, and AUC metrics. Table 8 shows the performance results from the second step of the second stage for Dataset 1, Dataset 2, and Dataset 3, based on the predefined gradient parameters. For Dataset 1, the experimental results achieved the highest score in confusion matrix metrics, with an accuracy of 0.99020, recall of 0.99546, and AUC score of 0.98891 for the gradient pair 0.30–0.40. However, the gradient pair 0.30–0.40 had the longest training time for this dataset, with 133.86 s. In contrast, the gradient pair 0.10–0.50 offered approximately 23.622% shorter training time with 0.00272 lower accuracy (0.98748) and a similar AUC (0.98664). From this perspective, using this parameter pair appears to be an efficient choice for balanced performance. In the experiments performed on Dataset 2, the gradient pair 0.10–0.50 demonstrated the highest performance in terms of the confusion matrix metrics. This configuration achieved scores of 0.98225, 0.99043, and 0.98234 for accuracy, recall, and AUC metrics, respectively. Additionally, it demonstrated performance comparable to the gradient pair 0.30–0.20, which hold the record for the shortest training time at 37.82 s, despite having a training time of 46.02 s. The results from Dataset 3 show

that different gradient pairs stand out with different metric performances. The gradient pair 0.30–0.20 achieved the highest accuracy score of 0.98642 for this particular metric. However, the gradient pair 0.10–0.50 achieved an accuracy score very close to the pair 0.30–0.20, with only a minimal difference of 0.0009. In the AUC metric, the gradient pair 0.10–0.50 achieved the highest performance with a score of 0.98499, albeit with a negligible difference (0.0001) compared with the gradient pair 0.30–0.20. Although the gradient pair 0.30–0.10 provided the highest recall with a score of 0.99449, it produced the lowest scores among all gradient pairs in terms of both accuracy and AUC metrics. The optimal gradient pair for this dataset was found to be 0.30–0.20 when assessing all performance metrics and training times. However, this pair significantly underperformed compared to the gradient pair 0.10–0.50 regarding accuracy, recall, and AUC metrics across other datasets. Therefore, for Dataset 3, the gradient pair 0.10–0.50 were identified as the optimal GOSS parameters, producing results quite close to the pair 0.30–0.20 in common performance.

Table 8. The performance results from the second step of the second stage for Dataset 1, Dataset 2, and Dataset 3, based on the predefined gradient parameters.

	GOSS Parameters			Performance Metrics			
	a	**b**	**Sum of Gradian Rates**	**Accuracy**	**Recall**	**AUC**	**Training Time (s)**
Dataset 1	0.30	0.40	0.70	0.99020	0.99546	0.98891	133.86
	0.10	0.50	0.60	0.98748	0.99091	0.98664	102.24
	0.30	0.20	0.50	0.98204	0.98636	0.98097	83.76
	0.30	0.10	0.40	0.98040	0.98364	0.97961	66.32
Dataset 2	0.30	0.40	0.70	0.97633	0.98804	0.97528	82.35
	0.10	0.50	0.60	0.98225	0.99043	0.98234	46.02
	0.30	0.20	0.50	0.97751	0.98804	0.97763	37.82
	0.30	0.10	0.40	0.96568	0.97847	0.96582	42.09
Dataset 3	0.30	0.40	0.70	0.98534	0.99089	0.98371	2152.10
	0.10	0.50	0.60	0.98552	0.98720	0.98499	1866.42
	0.30	0.20	0.50	0.98642	0.99168	0.98489	1494.26
	0.30	0.10	0.40	0.98419	0.99449	0.98181	1210.77

4.2. Experimental Studies

In experimental studies, the Light-ANFIS architecture, structured with gaussian membership functions, five membership sets, and a gradient pair of 0.10–0.50 as a result of ablation studies, was compared with the RFBFC-ANFIS architecture across three datasets. The experimental results were evaluated using confusion matrix metrics, training times, data sizes, and changes in accuracy and training errors according to epoch variations. Table 9 details the comparison between the proposed Light-ANFIS and RFBFC-ANFIS architectures for Dataset 1, Dataset 2, and Dataset 3 in terms of confusion matrix metrics.

In Dataset 1, the Light-ANFIS architecture appears to exhibit a performance behind the RFBFC-ANFIS architecture with an accuracy of 0.98748, precision of 0.98821, recall of 0.99091, F1-score of 0.98956, and AUC of 0.98664. However, the differences in scores are minimal. This difference is acceptable given the GOSS technique's role in reducing training data. In contrast, the proposed architecture outperformed RFBFC-ANFIS in Dataset 2, with an accuracy of 0.98225, precision of 0.97412, F1-score of 0.98221, and AUC of 0.98233. Regarding the recall metric, it achieved an equal score of 0.99043 compared to the RFBFC-ANFIS architecture. Furthermore, in Dataset 3, which has a larger data size, the proposed architecture outperformed RFBFC-ANFIS across all confusion matrix metrics. The Light-ANFIS architecture scored 0.98552 (accuracy), 0.98915 (precision), 0.98720 (recall), 0.98818 (F1-score), and 0.98503 (AUC). In comparison, the RFBFC-ANFIS architecture

achieved slightly lower scores of 0.98178, 0.98571, 0.98454, 0.98513, and 0.98097 for the same metrics.

Table 9. Comparison of the proposed Light-ANFIS architecture with the RFBFC-ANFIS architecture for Dataset 1, Dataset 2, and Dataset 3 in terms of confusion matrix metrics.

Datasets	Method	Accuracy	Precision	Recall	F1 Score	AUC
Dataset 1	RFBFC-ANFIS	0.99020	0.98827	0.99545	0.99185	0.98891
	Light-ANFIS	0.98748	0.98821	0.99091	0.98956	0.98664
Dataset 2	RFBFC-ANFIS	0.97988	0.96956	0.99043	0.97988	0.97999
	Light-ANFIS	0.98225	0.97412	0.99043	0.98221	0.98233
Dataset 3	RFBFC-ANFIS	0.98178	0.98571	0.98454	0.98513	0.98097
	Light-ANFIS	0.98552	0.98915	0.98720	0.98818	0.98503

Figure 5 illustrates the comparison between the Light-ANFIS and RFBFC-ANFIS architectures regarding the training times and data sizes. As shown in Figure 5a, the Light-ANFIS architecture achieved training times of 102.24 s, 46.02 s, and 1866.42 s for Dataset 1, Dataset 2, and Dataset 3, respectively, while the RFBFC-ANFIS architecture required 166.97 s, 77.81 s, and 3049.41 s for the same datasets. Converting the training time differences between the Light-ANFIS and RFBFC-ANFIS architectures into percentages indicates that the GOSS technique enhanced the Light-ANFIS efficiency by 38.77%, 40.86%, and 38.79% for these datasets. Based on the data used for training (Figure 5b), the analyses of both architectures show that Light-ANFIS requires fewer training samples than RFBFC-ANFIS across all datasets. The GOSS technique significantly enhanced Light-ANFIS by reducing the data sizes for RFBFC-ANFIS: Dataset 1 (4288→2573), Dataset 2 (1973→1184), and Dataset 3 (77,336→46,402).

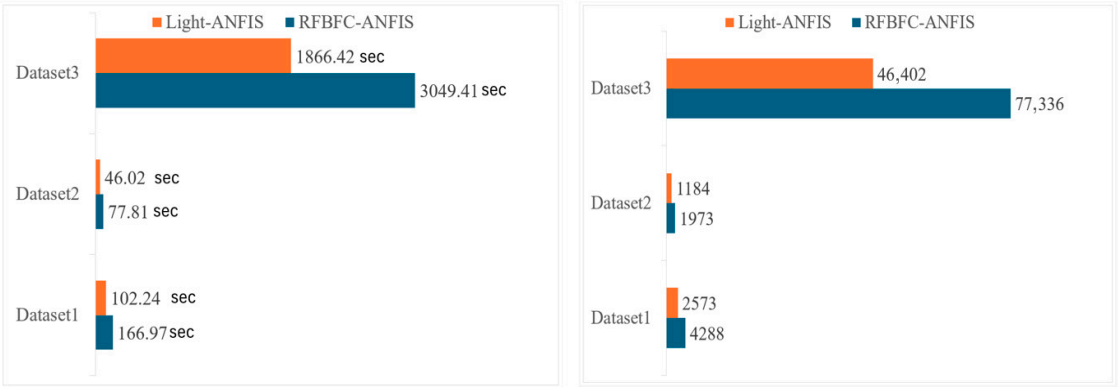
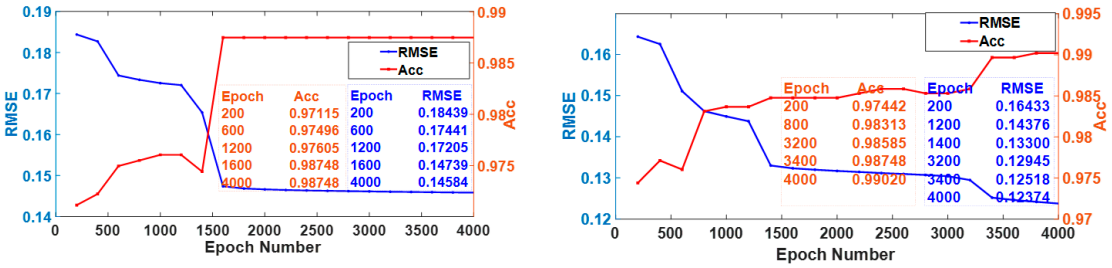


Figure 5. Comparison of Light-ANFIS and RFBFC-ANFIS architectures in terms of training times and data sizes: (**a**) Training times, (**b**) Training data sizes.

Comparisons addressing accuracy (Acc) and training errors (RMSE) based on epoch changes were made in experiments where epoch assignments were adjusted starting from 200 in 200-step intervals up to 4000. Figure 6 shows the changes in accuracy and RMSE scores according to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 1. In Figure 6a, Light-ANFIS architecture depicts a rapid decrease in the training error (RMSE) with changes in epochs ranging from 200 to 1600, accompanied by a proportional increase in accuracy. During this range, the RMSE score decreased from 0.18439 to 0.14739, while the accuracy improved from 0.97115 to 0.98748. For epoch changes in the 1600–4000 range, the accuracy rates stabilized at 0.98748, with only a slight reduction

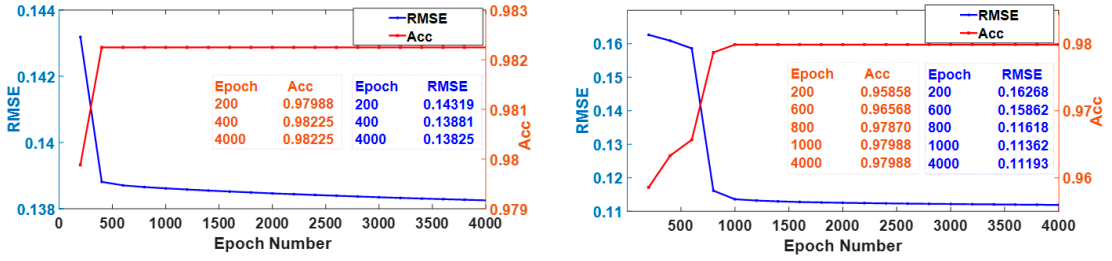
in RMSE (from 0.14739 to 0.14584). Figure 6b displays the graphs of the accuracy and RMSE scores of the RFBFC-ANFIS architecture as a function of the number of epochs. The changes in accuracy scores occurred faster in the epoch changes between 200 and 800, and the changes in RMSE scores occurred faster in the epoch changes between 200 and 1400. The accuracy increased from 0.97442 to 0.98313 in these ranges, whereas the RMSE decreased from 0.16433 to 0.13300. Beyond the 800th epoch, the accuracy continued to increase slowly, reaching 0.99020 by the 3800-th epoch. Similarly, after the 1400-th epoch, RMSE gradually declined to 0.12374 by the 4000-th epoch.



(a) (b)

Figure 6. Changes in accuracy and RMSE scores according to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 1: (**a**) Light-ANFIS, (**b**) RFBFC-ANFIS.

Figure 7 shows the accuracy and RMSE score changes relative to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 2. From the given graphs for the Light-ANFIS architecture (Figure 7a) rapid changes in accuracy and RMSE scores in the 200–400 epoch range were observed. In this range, the accuracy score increased linearly from 0.97988 to a peak of 0.98225 and remained stable until the 4000-th epoch. The RMSE score for the training error also showed a sharp decline from 0.14319 to 0.13881 in the above-mentioned range, followed by a minimal decrease to 0.13825 by the 4000-th epoch from this value. As shown in Figure 7b, similar to the Light-ANFIS architecture, the RFBFC-ANFIS architecture exhibited rapid changes in accuracy and RMSE scores between 200 and 800 epochs. In this range, the model accuracy increased from 0.95858 to 0.97870—very close to the peak accuracy—and achieved a peak value of 0.97988 in the next step (at 1000 epochs). The accuracy remained stable thereafter. Meanwhile, the RMSE score for the training error dropped sharply from 0.16268 to 0.11618 between 200 and 800 epochs, and then continued to decline slowly, reaching 0.11193 by the 4000-th epoch.



(a) (b)

Figure 7. Changes in accuracy and RMSE scores according to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 2: (**a**) Light-ANFIS, (**b**) RFBFC-ANFIS.

Figure 8 shows the graphs displaying the accuracy and RMSE changes according to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 3. The RMSE graphs for both architectures indicate an exponential reduction in the training errors in the 200–4000 epoch range. In the Light-ANFIS architecture, this decrease occurred in the range of 0.14636–0.13003 (Figure 8a), and in the RFBFC-ANFIS architecture,

it occurred in the range of 0.13561–0.12574 (Figure 8b). Across both architectures, the accuracy score changes in the 200–4000 epoch range were comparatively lower than those observed for the other datasets. In the Light-ANFIS architecture, the accuracy reached 0.98223 with epochs between 200 and 400, showing a minimal change (0.98280) in the 400–2200 epoch range. From this epoch value, it rose rapidly to reach 0.98540 score at the 2600-th epoch and then reached 0.98552 peak score in the next step, remaining stable until the 4000-th epoch. The RFBFC-ANFIS architecture exhibited fluctuating accuracy (0.98054–0.98126) between 200 and 1400 epochs, gradually rising afterward to reach its peak of 0.98178 at the 3800-th epoch.

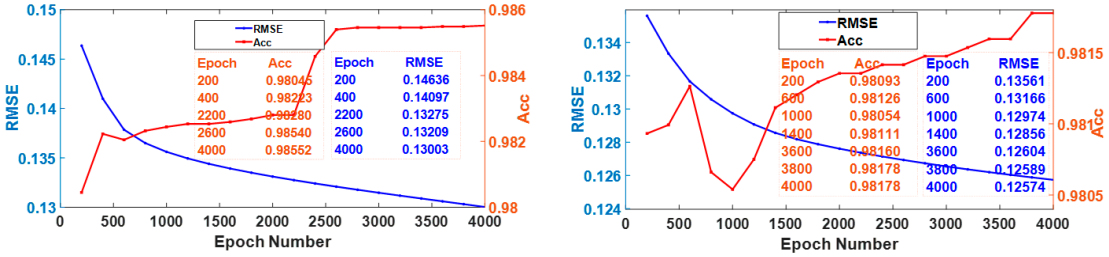


Figure 8. Changes in accuracy and RMSE scores according to the assigned epoch values in the Light-ANFIS and RFBFC-ANFIS architectures for Dataset 3: (**a**) Light-ANFIS, (**b**) RFBFC-ANFIS.

4.3. Literature Comparisons

When the literature is examined, it is observed that many studies use different methods and techniques for spam detection on social media platforms. This subsection compares the proposed architecture with existing literature, covering fuzzy inference system (FIS)-based approaches, current DL and ensemble learning (EL)-based techniques, and studies utilizing the same datasets. Table 10 presents a detailed performance comparison of related studies based on the methodology, dataset, and confusion matrix metrics.

Table 10. Performance comparison of the proposed study with related literature based on the methods and datasets.

No	Author(s)	Method(s)	Dataset	Acc (%)	Prec (%)	Rec (%)	F-Score (%)
1	Dhar and Bose [21]	Ensemble Models	Twitter Dataset 2	70.3	66.7	72.0	69.2
2	Atacak et al. [40]	Interval Type-2 Mamdani Fuzzy Inference System	Twitter Dataset 1	95.5	95.7	96.7	96.2
3	Li et al. [51]	Novel FL-driven web categorization system	Fake News Dataset	95.46	96.73	94.87	95.45
4	Suganthi and Prabha [55]	Fuzzy Similarity-based Hierarchical Clustering	Kaggle Social Media Dataset	92	-	-	-
5	Krishna and Srinivas [6]	Multi-modal fusion approach	Twitter Dataset	98.48	98.80	98.20	98.40
6	Laila et al. [85]	Unified Neuro-Fuzzy Inference System	Twitter and Amazon benchmark datasets	97.01	95.33	92.67	94.71
7	Ouni et al. [86]	BERT- and CNN-based TOBEAT approach	Twitter Dataset	94.97	94.05	95.88	94.95
8	Proposed Model	Light-ANFIS	Twitter Dataset 1	98.748	98.821	99.091	98.956
			Twitter Dataset 2	98.225	97.412	99.043	98.221
			Twitter Dataset 3	98.552	98.915	98.720	98.818

As can be seen from Table 10, two studies exist in the literature regarding the datasets used in this study. Dataset 1 was taken from a similar methodological study by Atacak et al. [40]. In the current study, the authors applied fuzzy logic (FL)-based methods, including Type-1 Mamdani (T1M-FIS), Type-1 Sugeno (T1S-FIS), Interval Type-2 Mamdani (IT2M-FIS), and Interval Type-2 Sugeno (IT2S-FIS) Fuzzy Inference Systems, to the Twitter data they collected and processed. They performed detailed analyses of the effectiveness and feasibility of these methods for detecting spam in social networks. Additionally, they compared the performance of these methods with four different ML methods. With experimental results of 95.5% accuracy, 95.7% precision, 96.7% recall, and 96.2% F-score, they achieved a successful performance using the IT2M-FIS. Although the DL-based EL method used in Dhan and Bose's study [21], from which part of Dataset 2 was derived, differs methodologically from our proposed approach; it shares functional similarities in achieving fast and acceptable performance in social networks. The researchers analyzed various models utilizing rich, naïve, and deep feature sets and introduced a novel ensemble approach that combined the CNN and LSTM models. This approach features intermediate accumulation, resolves time complexity without sacrificing performance, and enables real-time applications. Their method achieved prediction speeds 4–5 times faster than models relying on rich or deep feature sets. The 70.3% accuracy, 66.7% precision, 72.0% recall, and 69.2% F-score results obtained by the proposed model in terms of confusion matrix metrics indicate that the model achieves acceptable performance compared with the deep feature set-based model, which shows the highest performance in these metrics.

Since the architecture proposed in this study is methodologically based on the FL approach, as shown in the table, methods primarily relying on these techniques have been included. Li et al. [51] introduced an FL-based hybrid web categorization model for detecting fake news on social media. Their model employs DL-based methods for feature extraction and performs rule-based fake news detection using deep fuzzy learning (De_Fuz_Lear). Their experimental results—95.46% accuracy, 96.73% precision, 94.87% recall, and 95.45% F-score—proved that the proposed web categorization model can be successfully applied in this domain. Laila et al.'s Model [85], which extracts insights from tweets to determine user reliability based on promotion and spam scores, emerges as an effective and distinct method among current social media spam analysis approaches for achieving consequential context. By combining LSTM for score calculation and ANFIS for reliability assessment, the model achieved impressive results—97.01% accuracy, 95.33% precision, 92.67% recall, and 94.71% F-score—proving its effectiveness in the field. Suganthi and Prabha [55] proposed a fuzzy similarity-based hierarchical clustering method to detect accurate and hazardous communities in social networks relevant to this topic. For an effective categorization process, they applied a series of preprocessing steps, such as unigram, bigram, and 1–3 g tokenization, stop-word removal, stemming, TF-IDF, and word embedding for feature extraction on the data. In the subsequent process, dendrograms supported by fuzzy set matrices were obtained by analyzing the differences between the observed clusters. These dendrograms show the structure of the community using a transitive consensus matrix. The researchers achieved a high accuracy of 92% by applying this method to data obtained from Twitter.

The comparative studies listed in the table represent the current DL approaches. Among these approaches, transformer- and fusion-based DL methods are increasingly being adopted. Krishna and Srinivas proposed a robust multimodal fusion DL model called "StopSpamX" [6] to effectively identify spam content, which is a significant challenge on social media platforms. In the model configuration, word embedding techniques, including GloVe, Word2Vec, and FastText, and classifiers, including CNN, LSTM, Bi-LSTM, GRU, and CNN + Bi-LSTM fusion models, were used. The optimal performance was achieved

using the Word2Vec + CNN + Bi-LSTM fusion architecture, yielding an accuracy, precision, recall, and F-score of 98.48%, 98.80%, 98.20%, and 98.40%, respectively. Another effective DL-based approach for spam detection on Twitter was proposed by Ouni et al. [86]. The researchers named this model TOBEAT, which consists of a structure where tweets are embedded using the BERT method, and these results are combined with topic vectors before being classified through a CNN. The model demonstrated successful performance, achieving scores of 94.97%, 94.05%, 95.88%, and 94.95% for accuracy, precision, recall, and F-score metrics, respectively.

Comparing the results of the literature studies categorized above with those from the proposed Light-ANFIS architecture reveals that the proposed method outperforms existing BM- and DL-based approaches across all categories. When compared with the IT2M-FIS method by Atacak et al. (in the study where Dataset 1 was used), Light-ANFIS achieved a superior performance over IT2M-FIS by 3.248% in accuracy, 3.121% in precision, 2.391% in recall, and 2.756% in F-score. In contrast, the results from Dhan and Bose's study (using a subset of Dataset 2) lagged far behind the proposed architecture's performance on these metrics. Note, however, that the data in this case included numerous real-time features. When the performance of the proposed architecture is compared with that of the method proposed by Li et al., which gives the best performance in other metrics except the accuracy metric among the methods included in the category where FL-based approaches are addressed, it demonstrates superior performance across all datasets in these metrics. Specifically, in Dataset 1, where the proposed method achieved its best performance among other metrics except for the precision metric, it outperformed the new FL-based web of things categorization system, with improvements of 3.288%, 2.091%, 4.221%, and 3.506% in accuracy, precision, recall, and F-score, respectively. In the category of studies dealing with DL-based approaches that produce high-performance scores, the DL model proposed by Krishna and Srinivas, which achieved the highest performance, showed slightly better scores than our model in Dataset 2, except for the recall metric. However, the Light-ANFIS architecture surpassed the multimodal fusion approach across all metrics for Dataset 1 and Dataset 3. The proposed architecture achieved higher performance than the multimodal fusion approach in Dataset 1, with differences of 0.268%, 0.021%, 0.891%, and 0.556% in the accuracy, precision, recall, and F-score metrics, respectively. In Dataset 3, these differences were 0.072%, 0.115%, 0.52%, and 0.418%, respectively.

The model proposed in this study is advantageous because it does not require extensive datasets, resulting in reduced learning time and minimal resource consumption. Light-ANFIS offers a scalable solution with a high success rate in detecting spam content on social networks, low resource consumption, and high interpretability. In these aspects, it overcomes the limitations of existing methods in the literature and makes an innovative contribution to the field of spam detection.

Although the Light-ANFIS architecture cannot be directly used in real-time applications in its current structure, revealing the results regarding its computational efficiency is of critical importance in determining the roadmap for it in the future. The adequacy of the architecture in terms of computational efficiency for the relevant topic was evaluated through experiments measuring its parameter count, inference latency, and memory footprint, and comparing it with commonly used DL-based approaches in the literature. Table 11 presents the performance of the Light-ANFIS architecture in terms of computational efficiency obtained from the three datasets used in this study, alongside comparisons with commonly used CNN-LSTM and BERT-based methods in the literature.

Table 11. The performance assessment of the Light-ANFIS architecture in terms of computational efficiency, alongside comparisons with commonly used CNN-LSTM and BERT-based methods in the literature.

Model	Number of Parameters	Model Size (MB)	Inference Latency (ms)
Light-ANFIS (Proposed Model)	95	0.163	0.0031–0.0047
CNN-LSTM	~1–10 M	5–50	0.5–10
BERT-Base	~110 M	~1200	T4 GPU: 1–5

The ANFIS architecture represents a Sugeno-type first-order model with two inputs, five Gaussian membership sets, and one output. Because each Gaussian membership set had two parameters, the antecedent parameter was calculated as 20. This architecture uses 25 rules based on two inputs and five membership sets. Accordingly, the consequent (output) part consists of 75 parameters. Consequently, the simplified model architecture contained 95 parameters. Along with these parameters, the size of the model's weights and model format metadata in memory (serialized model size) was 0.163 MB. Experiments conducted on three datasets showed that the inference latency varied between 0.0031 and 0.0047 ms. This means that the simplified ANFIS architecture can process over 212,765 instances per second. These results demonstrate that ANFIS has a highly lightweight architecture in terms of parameter count and memory footprint, and that it is a high-speed model with inference latency. As summarized in Table 11, popular DL-based approaches in this field, such as CNN-LSTM [87,88] and BERT [89–91], are successful models, but they have high parameter counts and memory requirements. CNN-LSTM-based approaches typically have millions of parameters and require tens of MB of memory, while inference latency is also in the millisecond range. More advanced large-scale BERT models contain over a hundred million parameters, consume gigabytes of memory, and the inference latency can reach several milliseconds even on GPUs. Considering the computational efficiency findings, the Light-ANFIS architecture can be used in real-time social media analytics and similar online applications by integrating pre-processing steps that enable real-time data streaming and automatic feature extraction Via the Twitter API.

5. Conclusions

With the widespread use of the internet today, billions of users now spend time on social networking platforms. This increasing usage has made social networking sites vulnerable to malicious activities. Therefore, it has become an urgent necessity for these sites to take certain measures to ensure user security and protect their own reputation against malicious activities such as spam. Spam filters developed by social networking sites, such as X to protect their users are also inadequate because of the increasing number, variety, and behavioral changes in spam daily. Among the studies conducted so far based on the literature, most studies that achieved high accuracy in spam detection faced significant challenges due to their complex structures, resulting in lengthy training times and high computational costs. In this study, we introduced a novel spam detection approach called the Light-ANFIS architecture to address these issues. This method builds on a simplified Adaptive Network-based FIS and integrates RFBFC and GOSS techniques into the basic ANFIS framework. In the configuration of the proposed architecture, ablation studies were utilized to determine the optimal set type and count for ANFIS, as well as the values of the gradient parameter pair for the GOSS technique.

The experimental results were obtained by applying the Light-ANFIS and RFBFC-ANFIS architectures to three datasets. For this purpose, experiments were performed using these architectures to analyze the performance based on confusion matrix metrics, training

times, data sizes, and changes in accuracy and RMSE across different epochs. Confusion matrix analyses based on accuracy, precision, recall, F-score, and AUC metrics showed that the proposed Light-ANFIS architecture achieved high-performance scores across all three datasets. The comparative results between the RFBFC-ANFIS architecture (used to observe the effect of the GOSS technique on these metrics) and the proposed architecture demonstrate that the GOSS technique in the Light-ANFIS architecture can reduce the training data size without compromising performance. The experiments revealed that while the Light-ANFIS architecture showed a minor performance decline for Dataset 1 compared to RFBFC-ANFIS, for Dataset 2 and Dataset 3, it achieved slightly higher scores in all relevant metrics, even if minimal. The training time and data size analyses showed that the proposed architecture has shorter training times by 38.77%, 40.86%, and 38.79% for all three datasets, respectively, compared to the RFBFC-ANFIS architecture. This contribution is particularly necessary for large-scale data applications. Additionally, epoch-based analyses indicated that Light-ANFIS reached peak accuracy faster (with fewer epochs) than RFBFC-ANFIS for all datasets. Given the data sizes, larger datasets are expected to require more training epochs. When examining the results in terms of RMSE change, in Dataset 1 and Dataset 2 with a small data sizes, the minimal RMSE value was reached quickly (with a small epoch requirement), while in Dataset 3, which has more data sizes than both datasets, the minimal RMSE value was achieved later (with a more epoch requirement) through an exponential decline. Literature reviews indicate that the proposed Light-ANFIS architecture delivers competitive performance compared with high-scoring DL-based hybrid and fusion models. Moreover, with slight differences, it demonstrated a higher performance than the high-performance multimodal fusion model in all metrics for Dataset 1 and Dataset 3.

Dataset 3 used in this study consists of data we created ourselves, and a portion of the data was obtained from suspended accounts. The reasons for account suspension, such as spam content, hate speech, or manipulative behaviors, may disproportionately reflect certain user behaviors. Additionally, the demographic characteristics of these accounts, such as age, gender, and location, may be different from those of currently active accounts. This situation may lead to certain groups being over- or under-represented in the dataset and may limit the generalizability of the results. Therefore, the findings should be evaluated in this context.

The Light-ANFIS architecture, proposed as a high-performance spam detection method with a simple structural configuration, faces a key limitation: manual feature extraction during the data processing phase. This prevents the real-time implementation of the proposed architecture with its current structure. Instead of manual feature extraction in the data processing phase, adding a process that performs automatic feature extraction from data pulled Via the Twitter API could make the proposed architecture usable in real-time applications. In this sense, word embedding and advanced contextual embedding methods can be utilized. In our future study, we will focus on restructuring the data processing phase of the Light-ANFIS architecture as mentioned and developing a real-time spam detection model that operates based on data from the Twitter API.

Author Contributions: Conceptualization, O.Ç. and İ.A.; methodology, O.Ç. and İ.A.; software, O.Ç.; validation, O.Ç., İ.A. and İ.A.D.; formal analysis, O.Ç. and İ.A.; investigation, O.Ç. and İ.A.; resources, O.Ç., İ.A. and İ.A.D.; data curation, O.Ç. and İ.A.; writing—original draft preparation, O.Ç. and İ.A.; writing—review and editing, O.Ç., İ.A. and İ.A.D.; visualization, O.Ç.; supervision, İ.A. and İ.A.D.; project administration, İ.A. and İ.A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study employed three datasets: two publicly available datasets (Dataset 1 and Dataset 2) and a third dataset (Dataset 3) that we collected and processed. Dataset 3 is available upon request from the corresponding author. The raw, feature-extracted version of Dataset 1 can be found at https://peerj.com/articles/cs-1316/#supp-4 (accessed on 20 April 2025), while Dataset 2 in the same state is accessible via https://www.kaggle.com/datasets/whoseaspects/genuinefake-user-profile-dataset (accessed on 27 May 2025).

Acknowledgments: The authors would like to thank the Gazi University Academic Writing Application and Research Center for proofreading the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Acc	Accuracy
ACTM	Account Creation Time-Based Method
ADM	Anomaly Detection Method
AI	Artificial Intelligence
ANFIS	Adaptive Neuro-Fuzzy Inference System
AS	Automated Systems
AUC	Area Under ROC Curve
BAA	Behavioral Analysis Approaches
BERT	Bidirectional Encoder Representations from Transformers
BoW	The Bag of Words
CAA	Comparison and Contrastive Approaches
CF	Content-Based Filtering
CMBF	Combined Feature
CNN	Convolutional Neural Networks
COLAB	Google Colaboratory
D1	Dataset 1
D2	Dataset 2
D3	Dataset 3
DFS	Deep Feature Set
DIDM	Deceptive Information Detection Method
DL	Deep Learning
EFB	Exclusive Feature Bundling
ELM	Ensemble Learning Method
FFCM	Following and Follower Comparison Method
FIS	Fuzzy Inference Systems
FN	False Negative
FP	False Positive
FS-HC	Fuzzy Similarity-Based Hierarchical Clustering
GAT	Geolocation Analysis Technique
GD	Gradient Descent
GOSS	Gradient-Based One-Sided Sampling
GPT-3	Generative Pre-trained Transformer 3
GRU	Gated Recurrent Units
HSD	Honeypot-Based Spam Detection
IMDB	Internet Movie Database
IT2M-FIS	Interval Type-2 Mamdani Fuzzy Inference System
IT2S-FIS	Interval Type-2 Sugeno Fuzzy Inference System

LAA	Link Analysis Approach
LightGBM	Light Gradient Boosting Machine
LSE	Least Squares Estimation
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Machine Learning Methods
NFS	Naive Feature Set
NLP	Natural Language Processing
OOV	Out-Of-Vocabulary
Prec	Precision
Rec	Recall
RF	Random Forest
RFBFC	Random Forest-Based Feature Clustering
RFS	Rich Feature Set
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristics
SDT	Spammer Detection Tools
Spec	Specificity
SPO	Subject-Predicate-Object
T1M-FIS	Type-1 Mamdani Fuzzy Inference System
T1S-FIS	Type-1 Sugeno Fuzzy Inference System
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
T-TAM	Trend-Topics Analysis Method
UB	Using Blacklist

References

1. Patmanthara, S.; Febiharsa, D.; Dwiyanto, F.A. Social Media as a Learning Media: A Comparative Analysis of Youtube, WhatsApp, Facebook and Instagram Utillization. In Proceedings of the 2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE), Denpasar, Bali, Indonesia, 3–4 October 2019; Volume 6, pp. 183–186.
2. Masciantonio, A.; Bourguignon, D.; Bouchat, P.; Balty, M.; Rimé, B. Don't Put All Social Network Sites in One Basket: Facebook, Instagram, Twitter, TikTok, and Their Relations with Well-Being during the COVID-19 Pandemic. *PLoS ONE* **2021**, *16*, e0248384. [CrossRef]
3. Authenticity | X Help. Available online: https://help.x.com/en/rules-and-policies/authenticity (accessed on 15 June 2025).
4. Krithiga, R.; Ilavarasan, E. A Comprehensive Survey of Spam Profile Detection Methods in Online Social Networks. *J. Phys. Conf. Ser.* **2019**, *1362*, 012111. [CrossRef]
5. Mian, S.M.; Khan, M.S.; Shawez, M.; Kaur, A. Artificial Intelligence (AI), Machine Learning (ML) & Deep Learning (DL): A Comprehensive Overview on Techniques, Applications and Research Directions. In Proceedings of the 2024 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 10–12 July 2024; pp. 1404–1409.
6. Siva Krishna, D.; Srinivas, G. StopSpamX: A Multi Modal Fusion Approach for Spam Detection in Social Networking. *MethodsX* **2025**, *14*, 103227. [CrossRef]
7. Nasser, M.; Saeed, F.; Da'u, A.; Alblwi, A.; Al-Sarem, M. Topic-Aware Neural Attention Network for Malicious Social Media Spam Detection. *Alex. Eng. J.* **2025**, *111*, 540–554. [CrossRef]
8. Pal, A.A.; Mondal, S.; Kumar, C.A.; Kumar, C.J. A Transformer-Based Approach for Fake News and Spam Detection in Social Media Using RoBERTa. In Proceedings of the 2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI), Erode, India, 20–22 January 2025; pp. 1256–1263.
9. Çıtlak, O.; Dörterler, M.; Doğru, İ.A. A Survey on Detecting Spam Accounts on Twitter Network. *Soc. Netw. Anal. Min.* **2019**, *9*, 1–13. [CrossRef]
10. Choi, J.; Jeon, B.; Jeon, C. Scalable Learning Framework for Detecting New Types of Twitter Spam with Misuse and Anomaly Detection. *Sensors* **2024**, *24*, 2263. [CrossRef] [PubMed]
11. Rovito, L.; Bonin, L.; Manzoni, L.; De Lorenzo, A. An Evolutionary Computation Approach for Twitter Bot Detection. *Appl. Sci.* **2022**, *12*, 5915. [CrossRef]

12. Liu, S.; Wang, Y.; Zhang, J.; Chen, C.; Xiang, Y. Addressing the Class Imbalance Problem in Twitter Spam Detection Using Ensemble Learning. *Comput. Secur.* **2017**, *69*, 35–49. [CrossRef]
13. Fazil, M.; Abulaish, M. A Hybrid Approach for Detecting Automated Spammers in Twitter. *IEEE Trans. Inform. Forensic Secur.* **2018**, *13*, 2707–2719. [CrossRef]
14. Sánchez-Corcuera, R.; Zubiaga, A.; Almeida, A. Early Detection and Prevention of Malicious User Behavior on Twitter Using Deep Learning Techniques. *IEEE Trans. Comput. Soc. Syst.* **2024**, *11*, 6649–6661. [CrossRef]
15. Patel, P.; Bhushanwar, K.; Patel, H. Social Media Analysis for Criminal Behavior Detection: Methods, Application and Challenge. In Proceedings of the 2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL), Bhimdatta, Nepal, 18–20 February 2025; pp. 70–75.
16. Hussain, N.; Turab Mirza, H.; Rasool, G.; Hussain, I.; Kaleem, M. Spam Review Detection Techniques: A Systematic Literature Review. *Appl. Sci.* **2019**, *9*, 987. [CrossRef]
17. Li, C.; Liu, S. A Comparative Study of the Class Imbalance Problem in Twitter Spam Detection. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4281. [CrossRef]
18. Santos, I.; Miñambres-Marcos, I.; Laorden, C.; Galán-García, P.; Santamaría-Ibirika, A.; Bringas, P.G. Twitter Content-Based Spam Filtering. In Proceedings of the International Joint Conference SOCO'13-CISIS'13-ICEUTE'13, Salamanca, Spain, 11–13 September 2013; Herrero, Á., Baruque, B., Klett, F., Abraham, A., Snášel, V., de Carvalho, A.C.P.L.F., Bringas, P.G., Zelinka, I., Quintián, H., Corchado, E., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 449–458.
19. Maurya, S.K.; Singh, D.; Maurya, A.K. Deceptive Opinion Spam Detection Using Feature Reduction Techniques. *Int. J. Syst. Assur. Eng. Manag.* **2024**, *15*, 1210–1230. [CrossRef]
20. Dheyaa Radhi, A.; Obeid, H.N.; Al-Attar, B.; Fuqdan, A.-I.; Hakim, B.A.; Ali Hussein Al Naffakh, H. Unmasking Deceptive Profiles: A Deep Dive into Fake Account Detection on Instagram and Twitter. *BIO Web Conf.* **2024**, *97*, 00127. [CrossRef]
21. Dhar, S.; Bose, I. An Ensemble Deep Learning Model for Fast Classification of Twitter Spam. *Inf. Manag.* **2024**, *61*, 104052. [CrossRef]
22. El Mendili, F.; Fattah, M.; Berros, N.; Filaly, Y.; El Bouzekri El Idrissi, Y. Enhancing Detection of Malicious Profiles and Spam Tweets with an Automated Honeypot Framework Powered by Deep Learning. *Int. J. Inf. Secur.* **2024**, *23*, 1359–1388. [CrossRef]
23. Kumar, M.R.; Bharathi, P.S.; Sajiv, G. Accuracy Enhancement in Detection of Malicious Social Bots Using Reinforcement Learning Technique with URL Features in Twitter Network Through Convolutional Neural Network over K -Nearest Neighbors. In Proceedings of the 2024 Ninth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 4–5 April 2024; pp. 1–4.
24. Dhabliya, D.; Karthikeyan, C.; Sood, G.; Faiz, A.; Shah, M.D. Robust Twitter Spam Detection Through Ensemble Learning and Optimal Feature Selection. In Proceedings of the 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 13–14 December 2024; pp. 1–6.
25. Jeong, S.; Noh, G.; Oh, H.; Kim, C. Follow Spam Detection Based on Cascaded Social Information. *Inf. Sci.* **2016**, *369*, 481–499. [CrossRef]
26. Guo, D.; Chen, C. Detecting Non-Personal and Spam Users on Geo-Tagged Twitter Network. *Trans. GIS* **2014**, *18*, 370–384. [CrossRef]
27. Ong, Y.C.; Paladini, S.; Alifan, B.; Sambas, A.; Alwi, S.S.E.; Sedek, N.S.M. SohoNet: A Novel Social Honeynet Framework for Detecting Social Bots in Online Social Networks. *J. Adv. Res. Des.* **2024**, *1*, 234248. [CrossRef]
28. Krishna, C.R.; Loretta, G.I. An Efficient Malicious Social Bots with URL Features Detection Using Densenet Compared over ANN with Improved Accuracy. *AIP Conf. Proc.* **2025**, *3270*, 020151. [CrossRef]
29. Divani, N.; Vinitha, A. Machine Learning-Based Detection of Malicious URLs in Twitter. In Proceedings of the 2025 International Conference on Machine Learning and Autonomous Systems (ICMLAS), Prawet, Thailand, 11–13 March 2025; pp. 61–67.
30. Güngör, K.N.; Ayhan Erdem, O.; Doğru, İ.A. Tweet and Account Based Spam Detection on Twitter. In Proceedings of the The International Conference on Artificial Intelligence and Applied Mathematics in Engineering, Warsaw, Poland, 31 October–2 November 2025; Springer: Berlin/Heidelberg, Germany, 2020; pp. 898–905.
31. Asthana, Y.; Chhabra, R.; Srivastava, S. Machine Learning Techniques for Twitter Spam Detection: Comparative Insights and Real-Time Application. In Proceedings of the 2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 18–19 January 2024; pp. 780–786.
32. Asha, S.; Madhan, M.; PB, H.K.; Hariharan, S.; Dharaneesh, B. Twitter (X) Spam Detection Using Natural Language Processing by Encoder Decoder Model. In Proceedings of the 2024 1st International Conference on Sustainable Computing and Integrated Communication in Changing Landscape of AI (ICSCAI), Greater Noida, India, 4–6 July 2024; pp. 1–5.
33. Asmitha, M.; Kavitha, C.R. Exploration of Automatic Spam/Ham Message Classifier Using NLP. In Proceedings of the 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, 5–7 April 2024; pp. 1–7.

34. Gupta, S.; Khattar, A.; Gogia, A.; Kumaraguru, P.; Chakraborty, T. Collective Classification of Spam Campaigners on Twitter: A Hierarchical Meta-Path Based Approach. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 529–538.
35. Hwang, E.H.; Lee, S. A Nudge to Credible Information as a Countermeasure to Misinformation: Evidence from Twitter. *Inf. Syst. Res.* **2025**, *36*, 621–636. [CrossRef]
36. Nikhil Sai, G.V.; Tubagus, R.A.; Rohith, V.; Donavalli, H. Unlocking Deeper Data Insights on Social Media: Removing Hashtag and Tweets Spam for Improved Content Analysis. In Proceedings of the 2024 5th International Conference for Emerging Technology (INCET), Belgaum, India, 24–26 May 2024; pp. 1–6.
37. Gerber, A. A Content Analysis: Analyzing Topics of Conversation under the #sustainability Hashtag on Twitter. *Environ. Data Sci.* **2024**, *3*, e5. [CrossRef]
38. Inuwa-Dutse, I.; Liptrott, M.; Korkontzelos, I. Detection of Spam-Posting Accounts on Twitter. *Neurocomputing* **2018**, *315*, 496–511. [CrossRef]
39. Swe, M.M.; Nyein Myo, N. Fake Accounts Detection on Twitter Using Blacklist. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018; pp. 562–566.
40. Atacak, İ.; Çıtlak, O.; Doğru, İ.A. Application of Interval Type-2 Fuzzy Logic and Type-1 Fuzzy Logic-Based Approaches to Social Networks for Spam Detection with Combined Feature Capabilities. *PeerJ Comput. Sci.* **2023**, *9*, e1316. [CrossRef]
41. Kabakus, A.T.; Kara, R. A Survey of Spam Detection Methods on Twitter. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 29–38. [CrossRef]
42. Şencan, Ö.A.; Atacak, İ.; Doğru, İ.A. Sosyal Ağlarda Topluluk ve Konu Tespiti: Bir Sistematik Literatür Taraması. *Int. J. Inform. Technol.* **2022**, *15*.
43. Zhang, L.; Liu, W.; Wang, J. Design of Spam Detection and Classification System Based on Artificial Intelligence. In Proceedings of the 2025 5th International Symposium on Computer Technology and Information Science (ISCTIS), Xi'an, China, 16–18 May 2025; pp. 150–153.
44. Soto-Diaz, R.; Vásquez-Carbonell, M.; Escorcia-Gutierrez, J. A Review of Artificial Intelligence Techniques for Optimizing Friction Stir Welding Processes and Predicting Mechanical Properties. *Eng. Sci. Technol. Int. J.* **2025**, *62*, 101949. [CrossRef]
45. Shifath, S.M.S.-U.-R.; Khan, M.F.; Islam, M.S. A Transformer Based Approach for Fighting COVID-19 Fake News 2021. *arXiv* **2021**, arXiv:2101.12027. [CrossRef]
46. Alshattnawi, S.; Shatnawi, A.; AlSobeh, A.M.R.; Magableh, A.A. Beyond Word-Based Model Embeddings: Contextualized Representations for Enhanced Social Media Spam Detection. *Appl. Sci.* **2024**, *14*, 2254. [CrossRef]
47. Alom, Z.; Carminati, B.; Ferrari, E. A Deep Learning Model for Twitter Spam Detection. *Online Soc. Netw. Media* **2020**, *18*, 100079. [CrossRef]
48. Agarwal, R.; Dhoot, A.; Kant, S.; Singh Bisht, V.; Malik, H.; Ansari, M.F.; Afthanorhan, A.; Hossaini, M.A. A Novel Approach for Spam Detection Using Natural Language Processing with AMALS Models. *IEEE Access* **2024**, *12*, 124298–124313. [CrossRef]
49. Altwaijry, N.; Al-Turaiki, I.; Alotaibi, R.; Alakeel, F. Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models. *Sensors* **2024**, *24*, 2077. [CrossRef]
50. Wang, X.; Wang, K.; Chen, K.; Wang, Z.; Zheng, K. Unsupervised Twitter Social Bot Detection Using Deep Contrastive Graph Clustering. *Knowl.-Based Syst.* **2024**, *293*, 111690. [CrossRef]
51. Li, T.; Yu, J.; Zhang, H. Web of Things Based Social Media Fake News Classification with Feature Extraction Using Pre-Trained Convoluted Recurrent Network with Deep Fuzzy Learning. *Theor. Comput. Sci.* **2022**, *931*, 65–77. [CrossRef]
52. Wani, M.A.; ElAffendi, M.; Shakil, K.A. AI-Generated Spam Review Detection Framework with Deep Learning Algorithms and Natural Language Processing. *Computers (2073-431X)* **2024**, *13*, 264. [CrossRef]
53. Nair, V.; Pareek, J.; Bhatt, S. A Knowledge-Based Deep Learning Approach for Automatic Fake News Detection Using BERT on Twitter. *Procedia Comput. Sci.* **2024**, *235*, 1870–1882. [CrossRef]
54. Jain, D.K.; Kumar, A.; Sharma, V. Tweet Recommender Model Using Adaptive Neuro-Fuzzy Inference System. *Future Gener. Comput. Syst.* **2020**, *112*, 996–1009. [CrossRef]
55. Suganthi, R.; Prabha, K. Fuzzy Similarity Based Hierarchical Clustering for Communities in Twitter Social Networks. *Meas. Sens.* **2024**, *32*, 101033. [CrossRef]
56. Rajesh, K.P.; Nallasivam, M.P.; PS, S.P.; Kumar, H.; Dharun, V.S. Detection of Fake Hotel Reviews Using ANFIS and Natural Language Processing Techniques. In Proceedings of the 2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT), Kollam, India, 8–9 August 2024; Volume 1, pp. 265–269.
57. Gracia Betty, J.; Harivarthini, R.; Deepthi, O.; Pari, R.; Maharajan, P. YouTube Video Spam Comment Detection Using Light Gradient Boosting Machine. In Proceedings of the 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 3–5 August 2023; pp. 1650–1656.
58. Gong, D.; Liu, Y. A Mechine Learning Approach for Botnet Detection Using LightGBM. In Proceedings of the 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA), Changchun, China, 20–22 May 2022; pp. 829–833.

59. Aditya, B.L.; Mohanty, S.N. Heterogenous Social Media Analysis For Efficient Deep Learning Fake-Profile Identification. *IEEE Access* **2023**. [CrossRef]
60. Purba, K.R.; Asirvatham, D.; Murugesan, R.K. Classification of Instagram Fake Users Using Supervised Machine Learning Algorithms. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 2763. [CrossRef]
61. Çıtlak, O.; Doğru, İ.A.; Dörterler, M. Data Set Attributes Drawn in JSON Format on Twitter. *ResearchGate* **2018**. Available online: https://www.researchgate.net/publication/328655475_Data_set_attributes_drawn_in_JSON_format_on_Twitter (accessed on 13 June 2025).
62. Patro, S.G.K.; Sahu, K.K. Normalization: A Preprocessing Stage. *arXiv* **2015**, arXiv:1503.06462. [CrossRef]
63. Feature Selection Using Random Forest Classifier. Available online: https://www.geeksforgeeks.org/machine-learning/feature-selection-using-random-forest-classifier/ (accessed on 18 July 2025).
64. Zhao, Q.; Li, L.; Zhang, L.; Zhao, M. Recognition of Corrosion State of Water Pipe Inner Wall Based on SMA-SVM under RF Feature Selection. *Coatings* **2023**, *13*, 26. [CrossRef]
65. Feature Selection Using Random Forest GeeksforGeeks. Available online: https://www.geeksforgeeks.org/machine-learning/feature-selection-using-random-forest/ (accessed on 27 June 2025).
66. Gruber, P.; Agner, R.; Deniz, S. Detection of Cavitating States (Swirls) in a Francis Test Pump-Turbine Using Ultrasonic and Transient Pressure Measurements. In Proceedings of the 2018 12th International Group for Hydraulic Efficiency Measurements(IGHEM), Beijing, China, 10–13 September 2018; pp. 64–78.
67. Saha, S. Acoustic Assessment of Sleep Apnea and Pharyngeal Airway. Ph.D. Thesis, University of Toronto (Canada), Ontario, CA, Canada, 2021.
68. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
69. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3149–3157.
70. Feature Selection Techniques in Machine Learning GeeksforGeeks. Available online: https://www.geeksforgeeks.org/machine-learning/feature-selection-techniques-in-machine-learning/ (accessed on 26 August 2025).
71. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2020.
72. Shahani, N.M.; Zheng, X.; Guo, X.; Wei, X. Machine Learning-Based Intelligent Prediction of Elastic Modulus of Rocks at Thar Coalfield. *Sustainability* **2022**, *14*, 3689. [CrossRef]
73. Wang, R.; Liu, Y.; Ye, X.; Tang, Q.; Gou, J.; Huang, M.; Wen, Y. Power System Transient Stability Assessment Based on Bayesian Optimized LightGBM. In Proceedings of the 2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2), Changsha, China, 8–10 November 2019; pp. 263–268.
74. Al-Hmouz, A.; Shen, J.; Al-Hmouz, R.; Yan, J. Modeling and Simulation of an Adaptive Neuro-Fuzzy Inference System (ANFIS) for Mobile Learning. *IEEE Trans. Learn. Technol.* **2012**, *5*, 226–237. [CrossRef]
75. Adeyemo, Z.K.; Olawuyi, T.O.; Oseni, O.F.; Ojo, S.I. Development of a Path-Loss Prediction Model Using Adaptive Neuro-Fuzzy Inference System. *Int. J. Wirel. Microw. Technol.* **2019**, *9*, 40–53.
76. Jang, J.-S.R. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
77. Dağkurs, B.; Atacak, İ. Deep Learning-Based Novel Ensemble Method with Best Score Transferred-Adaptive Neuro Fuzzy Inference System for Energy Consumption Prediction. *PeerJ Comput. Sci.* **2025**, *11*, e2680. [CrossRef]
78. Turk, F. RNGU-NET: A Novel Efficient Approach in Segmenting Tuberculosis Using Chest X-Ray Images. *PeerJ Comput. Sci.* **2024**, *10*, e1780. [CrossRef]
79. Gharaibeh, M.; Almahmoud, M.; Ali, M.Z.; Al-Badarneh, A.; El-Heis, M.; Abualigah, L.; Altalhi, M.; Alaiad, A.; Gandomi, A.H. Early Diagnosis of Alzheimer's Disease Using Cerebral Catheter Angiogram Neuroimaging: A Novel Model Based on Deep Learning Approaches. *Big Data Cogn. Comput.* **2022**, *6*, 2. [CrossRef]
80. Liu, X.; Lu, H.; Nayak, A. A Spam Transformer Model for SMS Spam Detection. *IEEE Access* **2021**, *9*, 80253–80263. [CrossRef]
81. Valero-Carreras, D.; Alcaraz, J.; Landete, M. Comparing Two SVM Models through Different Metrics Based on the Confusion Matrix. *Comput. Oper. Res.* **2023**, *152*, 106131. [CrossRef]
82. Understanding the Confusion Matrix in Machine Learning. Available online: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (accessed on 21 June 2025).
83. Atacak, İ. An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection. *Appl. Sci.* **2023**, *13*, 1484. [CrossRef]
84. Jayaswal, V. Performance Metrics: Confusion Matrix, Precision, Recall, and F1 Score. *Towards Data Sci.* **2020**.
85. Laila, K.; Jayashree, P.; Vinuvarsidh, V. A Unified Neuro-Fuzzy Framework to Assess the User Credibility on Twitter. *IETE J. Res.* **2024**, *70*, 1407–1424. [CrossRef]
86. Ouni, S.; Fkih, F.; Omri, M.N. BERT- and CNN-Based TOBEAT Approach for Unwelcome Tweets Detection. *Soc. Netw. Anal. Min.* **2022**, *12*, 144. [CrossRef]

87. Airlangga, G.; Bata, J.; Adi Nugroho, O.I.; Lim, B.H.P. Hybrid CNN-LSTM Model with Custom Activation and Loss Functions for Predicting Fan Actuator States in Smart Greenhouses. *AgriEngineering* **2025**, *7*, 118. [CrossRef]
88. Zhang, Y.; Zhang, P.; Zhang, W.; Wang, M. CNN-LSTM-Attention with PSO Optimization for Temperature and Fault Prediction in Meat Grinder Motors. *Discov. Appl. Sci.* **2025**, *7*, 438. [CrossRef]
89. BERT Inference on G4 Instances Using Apache MXNet and GluonNLP: 1 Million Requests for 20 Cents | Artificial Intelligence. Available online: https://aws.amazon.com/blogs/machine-learning/bert-inference-on-g4-instances-using-apache-mxnet-and-gluonnlp-1-million-requests-for-20-cents/ (accessed on 31 August 2025).
90. Performance Regression Found in TensorRT 8.6.1 When Running BERT on GPU T4 Deep Learning (Training & Inference)/TensorRT. Available online: https://forums.developer.nvidia.com/t/performance-regression-found-in-tensorrt-8-6-1-when-running-bert-on-gpu-t4/261651 (accessed on 31 August 2025).
91. What Differences in Inference Speed and Memory Usage Might You Observe Between Different Sentence Transformer Architectures (for Example, BERT-Base vs DistilBERT vs RoBERTa-Based Models)? Available online: https://milvus.io/ai-quick-reference/what-differences-in-inference-speed-and-memory-usage-might-you-observe-between-different-sentence-transformer-architectures-for-example-bertbase-vs-distilbert-vs-robertabased-models (accessed on 31 August 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.